COPPE/UFRJ

ALGORITMOS POLINOMIAIS PARA PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA COM MULTI-OBJETIVOS EM GRAFOS

Leizer de Lima Pinto

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Nelson Maculan Filho

Rio de Janeiro

Dezembro de 2009

ALGORITMOS POLINOMIAIS PARA PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA COM MULTI-OBJETIVOS EM GRAFOS

Leizer de Lima Pinto

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Nelson Maculan Filho, D. Habil.

Prof. Cláudio Thomás Bornstein, Dr. Rer. Nat.

Prof. Paolo Toth, Ph.D.

Profa. Celina Miraglia Herrera de Figueiredo, D.Sc.

Prof. Abilio Pereira de Lucena Filho, Ph.D.

Profa. Nair Maria Maia de Abreu, D.Sc.

Prof. Geraldo Robson Mateus, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2009

Pinto, Leizer de Lima

Algoritmos polinomiais para problemas de otimização combinatória com multi-objetivos em grafos/Leizer de Lima Pinto. – Rio de Janeiro: UFRJ/COPPE, 2009.

IX, 77 p. 29, 7cm.

Orientador: Nelson Maculan Filho

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2009.

Referências Bibliográficas: p. 29 – 33.

1. Problemas Multi-Objetivo. 2. Funções Gargalo. 3. Solução Pareto-Ótima. I. Maculan Filho, Nelson. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao meu avô Benigno

*(*in memoriam*).*

Agradecimentos

Primeiramente agradeço a Deus pela minha existência e pelos orientadores atenciosos, respeitosos e éticos que tive nesta minha caminhada; aos Professores Cláudio Bornstein e Nelson Maculan que me proporcionaram um ótimo ambiente de trabalho, pela orientação, ensinamento, atenção, compreensão e pelo apoio de sempre; aos Professores Celina Figueiredo, Nair Abreu, Abilio Lucena, Geraldo Mateus e Paolo Toth por terem aceito participar desta avaliação; ao Professor Gilbert Laporte da HEC/Universidade de Montreal pela presteza e atenção durante o período em que estive no CIRRELT (de janeiro a julho de 2009), e pelo trabalho em conjunto no desenvolvimento do algoritmo para um problema tri-objetivo de árvore de Steiner, o qual compõe parte desta tese; à Professora Marta Pascoal do INESC/Universidade de Coimbra pelas nossas discussões, em janeiro de 2009 no CIRRELT, que resultaram nos algoritmos Escada e Blocos para um problema de caminho tri-objetivo, os quais também compõem parte desta tese; ao Professor Marco Antonio da Universidade Católica de Goiás pelos ensinamentos durante os anos de iniciação científica, os quais foram fundamentais para os meus estudos aqui na COPPE; ao meu amigo e irmão (de coração) Elivelton, pela amizade ao longo destes anos, pela contribuição no meu ingresso aqui na COPPE, e pela recepção e companhia em Montreal; aos companheiros do G4 e grandes amigos, Fábio, Jesus e Vinícius, que estão sempre prontos para ajudar, pelos seminários, pelo futebol e pelos vários momentos de descontração; à Fátima, à Taísa e à todas as secretárias do PESC pela ajuda de sempre; à minha mãe, ao meu pai, à minha irmã e ao meu sobrinho Fellipe pelo amor, carinho e por compreenderem minha ausência em momentos importantes de suas vidas; às minhas tias, tios, primas e primos que, mesmo distantes, estão sempre torcendo pelas minhas conquistas; ao amigo Dr Siguero Taia pela motivação de sempre e pela agradável pescaria no rio Araguaia em agosto de 2009; aos meus

amigos da minha cidade natal, Itaguaru-GO, que sempre me recebem tão bem; ao Conselho Nacional de Desenvolvimento Científico e Tecnológico do Brasil (CNPq) pelo suporte financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALGORITMOS POLINOMIAIS PARA PROBLEMAS DE OTIMIZAÇÃO COMBINATÓRIA COM MULTI-OBJETIVOS EM GRAFOS

Leizer de Lima Pinto

Dezembro/2009

Orientador: Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

O foco deste trabalho são problemas de otimização combinatória multi-objetivo em grafos considerando, no máximo, uma função totalizadora (por exemplo, Min-Sum ou MaxProd). As demais funções objetivo consideradas são do tipo gargalo (MinMax ou MaxMin). Algoritmos polinomiais para a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas serão apresentados, juntamente com demonstrações de otimalidade, experimentos computacionais e aplicações para o caso tri-objetivo. Em particular, árvore geradora, árvore de Steiner e caminho são os problemas considerados nesta tese.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

POLYNOMIAL ALGORITHMS FOR MULTI-OBJECTIVE COMBINATORIAL OPTIMIZATION PROBLEMS IN GRAPHS

Leizer de Lima Pinto

December/2009

Advisor: Nelson Maculan Filho

Department: Systems Engineering and Computer Science

The focus of this work are multi-objective combinatorial optimization problems in graphs with at most one totalizing objective function (for example, MinSum or MaxProd). The other objectives are of the bottleneck type (MinMax or MaxMin). Polynomial algorithms are developed which generate a minimal complete set of Pareto-optimal solutions. Optimality proofs, computational experiments and applications for tri-objective cases are given. In particular, spanning tree, Steiner tree and optimal paths problems are considered.

Sumário

Capítulo 1

Introdução e Revisão Bibliográfica

Nesta tese estamos interessados em problemas de Otimização Combinatória Multi-objetivo em Grafos (OCMG). Algoritmos polinomiais para estes problemas serão apresentados, que são os resultados principais deste trabalho.

Na resolução de problemas mono-objetivo a meta principal é a obtenção de uma solução ótima. Algoritmos para problemas de caminho e de árvore geradora são apresentados em Ahuja *et al* [1]. Um software didático para oito problemas de caminho mono-objetivo incluindo MinSum, MinMax e MaxProd é apresentado em Pinto e Bornstein [26], contendo implementações dos algoritmos de Dijkstra [11] e de atualização de distâncias (veja, por exemplo, Bellman [3]).

Em problemas multi-objetivo a idéia de solução ótima é substituída pela idéia de solução Pareto-ótima (também podemos dizer solução eficiente ou solução não dominada) e o interesse é obter um conjunto mínimo (ou máximo) completo de soluções Pareto-ótimas.

As primeiras definições de dominância e solução Pareto-ótima surgiram em 1896 com Pareto [24]. O primeiro trabalho na área da Pesquisa Operacional (PO) utilizando esses conceitos foi apresentado por Koopmans [21] em 1951. Esses conceitos são fundamentais no campo da Programação multi-objetivo, que é estudada em diversas áreas da PO, por exemplo em Programação Linear (veja Zeleny [39]).

Normalmente, dois tipos de funções objetivo são consideradas na literatura de

OCMG: soma e gargalo. Estas funções avaliam as soluções através da soma dos pesos de seus elementos (MinSum) e pelo pior peso de seus elementos (MinMax ou MaxMin), respectivamente. Denominamos essas funções que envolvem todos os elementos da solução, como as do tipo soma, de *funções totalizadoras*. A função MaxProd, que consiste em maximizar o produto dos pesos dos elementos da solução, é um outro exemplo de função totalizadora. Ehrgott e Gandibleux [12] apresentam mais de duzentas referências em otimização combinatória multi-objetivo.

Hansen [18], Martins [23] e Berman *et al* [4] apresentam algoritmos polinomiais para o problema de caminho bi-objetivo com uma função totalizadora e outra de gargalo. Aqui continuaremos considerando apenas uma função totalizadora. Além disso, também consideramos outros problemas de Otimização Combinatória. Esta restrição de termos no máximo uma função objetivo do tipo totalizadora se deve ao fato que a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas para os problemas de OCMG (caminho, designação e árvore geradora) é NP-difícil (veja Serafini [36] e Camerini *et al* [5]).

Algoritmos não polinomiais para o problema de caminho multi-objetivo com mais de uma função MinSum são apresentados em Martins [22], Azevedo e Martins [2], Tung e Chew [38], Guerriero e Musmanno [17] e Gandibleux *et al* [15]. Além das funções MinSum, uma função MaxMin é considerada neste último trabalho.

Para a obtenção de uma árvore geradora MinMax em grafos orientados, Gabow e Tarjan [14] apresentam um algoritmo mono-objetivo o qual contém um mecanismo de inserção de arcos ao longo de suas iterações. O algoritmo tenta encontrar uma solução em grafos parciais do grafo original, os quais são definidos impondo restrições nos pesos dos arcos. Este mecanismo de inserção de arcos também é utilizado pelo algoritmo MMS de Pinto *et al* [28] para o problema tri-objetivo MaxMin-MinMax-MinSum. Um outro algoritmo, MMS-R, desenvolvido por Pinto *et al* [29] para este problema tri-objetivo, trabalha com uma idéia reversa ao MMS, isto é, com uma mecânica de deleção de arcos ao longo das iterações. O mesmo procedimento é utilizado no algoritmo de Berman *et al* [4] para o problema bi-objetivo MinMax-MinSum. A generalização do MMS-R para o caso multi-objetivo com uma função

totalizadora e as demais de gargalo foi apresentado em Pinto *et al* [31].

O MMS-R foi desenvolvido com a intenção de reduzir o número de iterações gastas pelo MMS. Continuando com esta intenção, dois novos algoritmos tri-objetivo baseados no MMS-R, MMS-S e MMS-B, foram desenvolvidos por Pinto e Pascoal [33, 34], juntamente com uma variante de cada um. Comparando com os demais métodos, o número de iterações executadas por MMS-B (e sua variante) é sempre menor ou igual, consequentemente o tempo computacional consumido também é menor. Em Pinto e Pascoal [34] um algoritmo de rotulação, que é uma extensão do algoritmo de Dijkstra para o problema de caminho tri-objetivo com duas funções objetivo MinMax e uma MinSum, também foi desenvolvido e comparado com os métodos citados acima. Nestas comparações o algoritmo MMS-B foi o mais eficiente em todas as instâncias resolvidas. Como teoricamente a complexidade é a mesma para todos os métodos exceto para o algoritmo de rotulação, que tem complexidade pior, então podemos dizer que o MMS-B é melhor do que os demais. Por este motivo, aqui iremos generalizar este método para o caso multi-objetivo, assim como foi feito em [31] para o MMS-R.

Um algoritmo polinomial para um problema de árvore de Steiner contendo três objetivos foi desenvolvido por Pinto e Laporte [32]. As funções objetivo deste problema consistem em maximizar o retorno financeiro e minimizar a máxima distância entre cada par de nós interconectados, bem como o máximo número de arcos entre a raiz e cada nó. Aplicações para este problema aparecem no planejamento de redes de telecomunicações para acesso local, como por exemplo, na construção de uma rede de fibra ótica para disponibilizar conexão de Internet banda larga em edifícios comerciais e residenciais. Outros problemas de árvore de Steiner NP-difíceis com aplicações nesta área são estudados em [6, 8, 10, 19]. Zelikovsky [40] apresenta aplicações associadas com o problema mono-objetivo de árvore de Steiner de gargalo.

No Capítulo 2 são apresentados os problemas juntamente com definições e aplicações. No Capítulo 3 são apresentados algoritmos para o problema de caminho tri-objetivo com duas funções de gargalo e uma totalizadora. No Capítulo 4 é apresentado um algoritmo para um problema tri-objetivo de árvore de Steiner. No

Capítulo 5 são apresentados algoritmos multi-objetivo. No Capítulo 6, finalmente, apresentam-se as considerações finais sobre o trabalho.

Capítulo 2

Problemas

Nosso objetivo neste capítulo é definir um problema de otimização combinatória multi-objetivo em grafos com $l+1$ objetivos, composto por l funções MinMax e uma função MinSum. Além disso, é definido um problema tri-objetivo de árvore de Steiner. Definições e aplicações associadas aos problemas também são apresentadas neste capítulo.

2.1 Um modelo multi-objetivo para caminhos e árvore geradora

Considere o digrafo $G = (N, M)$, onde N é o conjunto dos nós e M o conjunto de arcos, com $|N| = n$ e $|M| = m$. Seja l um inteiro positivo limitado superiormente por alguma constante $\lambda \in \mathbb{Z}$. Para cada $(i, j) \in M$ associamos as funções reais $p^k(i, j) \in \mathbb{R}$, $k = 1, 2, \ldots, l+1$. Os $p^k(i, j)$ representam os pesos associados às $l+1$ funções objetivo. Sejam $p^k_1, p^k_2, \ldots, p^k_{m_k}$ os diferentes valores assumidos por $p^k(.)$ em M, $k = 1, 2, \ldots, l$. Temos, então, que $m_k \leq m$ é a quantidade de valores distintos assumidos por $p^k(.)$. Sem perda de generalidade, suponhamos que:

$$p^k_1 > p^k_2 > \cdots > p^k_{m_k}, \quad k = 1, 2, \ldots, l. \tag{2.1}$$

Estas ordenações feitas acima para os valores assumidos por $p^k(.), k = 1, 2, \ldots, l$,

são fundamentais para os algoritmos que foram desenvolvidos. Para $p^{l+1}(.)$ nenhuma ordenação é necessária.

O que também é fundamental para os algoritmos desenvolvidos é o grafo parcial de G que será apresentado agora. Seja $v \in \mathbb{R}^l$, $v = (v_k)$, tal que $1 \leq v_k \leq m_k$, para $k = 1, 2, \ldots, l$, um vetor de índices. Considere o subconjunto de arcos de M, $M_v = \{(i,j) \in M \mid p^k(i,j) \leq p^k_{v_k}, k = 1, 2, \ldots, l\}$. O grafo parcial $G_v = (N, M_v)$ de G é utilizado em cada iteração de nossos algoritmos. Observe que se v é tal que $v_k = 1, \forall k \in \{1, 2, \ldots, l\}$ então temos $G_v = G$.

Seja $s \subseteq M$ uma solução viável satisfazendo alguma propriedade α. Por exemplo, s pode ser uma árvore geradora ou um caminho entre dois nós de G. Seja S o conjunto de todas as soluções viáveis. Segue-se a formulação de um problema com uma função totalizadora do tipo MinSum e l funções gargalo do tipo MinMax.

$$
\begin{array}{lll}
(P) & \textit{minimizar} & \max\limits_{(i,j)\in s}\{p^k(i,j)\}, \quad k = 1, 2, \ldots, l \\
 & \textit{minimizar} & \sum\limits_{(i,j)\in s} p^{l+1}(i,j) \\
 & \textit{sujeito a:} & s \in S.
\end{array}
$$

As l primeiras funções objetivo foram consideradas do tipo MinMax apenas por simplicidade. Para estas l funções podemos considerar qualquer combinação entre MinMax e MaxMin, pois estas são equivalentes (veja, por exemplo, Hansen [18]).

Seguem-se definições associadas ao problema (P) que estaremos utilizando no decorrer deste trabalho.

Definições 2.1.1 *Considere o problema (P). Dizemos que:*

(a) $q^s \in \mathbb{R}^{l+1}$, $q^s = (q^s_k)$, é o ***vetor objetivo*** *associado à solução $s \in S$, onde*

$$q^s_k = \max_{(i,j)\in s}\{p^k(i,j)\}, k = 1, 2, \ldots, l \quad e \quad q^s_{l+1} = \sum_{(i,j)\in s} p^{l+1}(i,j).$$

(b) $s \in S$ ***domina*** *$\bar{s} \in S$ quando $q^s \leq q^{\bar{s}}$ com $q^s \neq q^{\bar{s}}$.*

(c) $s \in S$ *é um* ***Pareto-ótimo*** *quando não existe* $\bar{s} \in S$ *que domina* s.

(d) Um conjunto de soluções Pareto-ótimas $S^* \subseteq S$ *é um* ***conjunto mínimo completo*** *quando:*

(d_1*)* $\forall s^1, s^2 \in S^*$ *temos* $q^{s^1} \neq q^{s^2}$*; e,*

(d_2*) Para qualquer Pareto-ótimo* $s \in S$ *existe* $\bar{s} \in S^*$ *tal que* $q^{\bar{s}} = q^s$.

(e) O conjunto de todas as soluções Pareto-ótimas é o ***conjunto máximo completo***.

Um dos principais objetivos aqui é apresentar algoritmos para a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas para o problema (P). Observe que pode existir mais de um conjunto desse tipo, mas todos com o mesmo número de soluções.

Vejamos no resultado que se segue que a cardinalidade de um conjunto mínimo completo é limitada polinomialmente pelos dados de entrada.

Proposição 2.1.2 *Seja* S^* *um conjunto mínimo completo de soluções Pareto-ótimas para o problema* (P). *Temos que* $|S^*| \leq \prod_{k=1}^{l} m_k \leq m^l$.

Demonstração: Suponha, sem perda de generalidade, que existe alguma solução viável, ou seja, $S \neq \emptyset$. Seja $s \in S$ uma solução Pareto-ótima qualquer para (P). Obviamente que $q_k^s = p_{v_k}^k$, $k = 1, 2, \ldots, l$, para algum vetor de índices $v \in \mathbb{R}^l$ tal que $1 \leq v_k \leq m_k$. Para qualquer outra solução $\bar{s} \in S$ com $q_k^{\bar{s}} = p_{v_k}^k = q_k^s$, $\forall k \in \{1, 2, \ldots, l\}$ temos $q_{l+1}^{\bar{s}} \geq q_{l+1}^s$, pois s é um Pareto-ótimo. Deste modo, pela definição de conjunto mínimo completo, para cada vetor de índices v com $1 \leq v_k \leq m_k$, $k = 1, 2, \ldots, l$, temos no máximo uma solução em S^*. O número de possibilidades para v é $\prod_{k=1}^{l} m_k$, portanto, $|S^*| \leq \prod_{k=1}^{l} m_k \leq m^l$. ∎

Um problema particular de árvore de Steiner contendo três objetivos é apresentado na próxima seção.

2.2 Um problema de árvore de Steiner

Considere o grafo $G = (N, M)$ definido na seção anterior. Aqui, para cada arco $(i, j) \in M$ associamos apenas uma distância $d(i, j) \in \mathbb{R}$ e, para cada nó $i \in N$, associamos um retorno financeiro $r(i) \geq 0$. Dado um conjunto de nós $T \subseteq N$, uma árvore direcionada $A = (V, E)$ de G com raiz $s \in T$ é chamada de *árvore de Steiner* quando $V \supseteq T$. Dizemos que T é o *conjunto de nós terminais*. Seja $h(i)$ o número de arcos, em A, entre a raiz s e o nó i, $\forall i \in V$.

Aqui o conjunto viável S é o conjunto de todas as árvores de Steiner em G. Segue-se o problema tri-objetivo de árvore de Steiner o qual estamos interessados neste trabalho:

$$
\begin{array}{lll}
(P_1) & \textit{minimizar} & \max\limits_{i \in V}\{h(i)\} \\
 & \textit{minimizar} & \max\limits_{(i,j) \in E}\{d(i,j)\} \\
 & \textit{maximizar} & \sum\limits_{i \in V} r(i) \\
 & \textit{sujeito a:} & A = (V, E) \in S.
\end{array}
$$

Definições 2.2.1 *Considere o problema* (P_1). *Dizemos que:*

(a) $f(A) \in \mathbb{R}^3$, $f(A) = (f_1(A), f_2(A), f_3(A))$, *é o* ***vetor objetivo*** *associado à solução* $A = (V, E) \in S$, *onde*

$$f_1(A) = \max_{i \in V}\{h(i)\}, \quad f_2(A) = \max_{(i,j) \in E}\{d(i,j)\} \quad e \quad f_3(A) = \sum_{i \in V} r(i).$$

(b) $A \in S$ ***domina*** $\bar{A} \in S$ *quando* $f_k(A) \leq f_k(\bar{A})$, $k = 1, 2$, *e* $f_3(A) \geq f_3(\bar{A})$ *com* $f(A) \neq f(\bar{A})$.

(c) $A \in S$ *é um* ***Pareto-ótimo*** *quando não existe* $\bar{A} \in S$ *que domina* A.

(d) Um conjunto de soluções Pareto-ótimas $S^* \subseteq S$ *é um* ***conjunto mínimo completo*** *quando:*

(d_1) $\forall \bar{A}, \hat{A} \in S^*$ temos $f(\bar{A}) \neq f(\hat{A})$; e,

(d_2) Para qualquer Pareto-ótimo $A^* \in S$ existe $A \in S^*$ tal que $f(A) = f(A^*)$.

Na próxima seção apresentamos duas aplicações para o problema (P) e uma para o problema (P_1).

2.3 Aplicações

Para as duas primeiras aplicações apresentadas aqui vamos considerar o problema de caminho tri-objetivo MaxMin-MinMax-MinSum. Temos que a função MaxMin pode ser transformada na função MinMax (basta multiplicar os pesos dos arcos associados a esta função por -1). Então o problema MaxMin-MinMax-MinSum é equivalente ao problema (P) com $l = 2$. A última aplicação apresentada aqui se refere ao problema de Steiner (P_1).

2.3.1 Um problema de transporte

Considere um empresa que trabalha com o transporte terrestre de passageiros entre cidades. Para cada viagem que ela deve realizar de uma cidade r até outra cidade t ela necessita decidir qual será o caminho, ou seja, quais os trechos percorridos. Cada arco (i, j) representa a estrada que vai da cidade i até a cidade j. Pressupomos que existe um único arco indo de i até j. Para cada arco existem três pesos associados, p^1, p^2 e p^3 tais que:

- $p^1(i, j)$ mede a qualidade da estrada representada pelo arco (i, j). Quanto menor, pior é a qualidade;
- $p^2(i, j)$ mede o congestionamento da estrada representada pelo arco (i, j). Quanto maior, mais congestionada é a estrada;
- $p^3(i, j)$ é o custo de percorrer a estrada representada pelo arco (i, j). Pode, por exemplo, ser proporcional à quilometragem percorrida.

O fato de uma estrada ser muito ruim pode implicar em quebras, perdas, atrasos e até mesmo inviabilizar a viagem. O congestionamento pode, por exemplo, causar muita insatisfação aos passageiros, além de aumentar o tempo da viagem. Além disso, existem os custos com gasolina, pneus, óleo, etc. A empresa em questão pode estar interessada no problema de encontrar um caminho de r a t tal que:

- O pior trecho do caminho seja o melhor possível (MaxMin em p^1);
- O trecho mais congestionado do caminho tenha o menor congestionamento possível (MinMax em p^2);
- O custo pago para percorrer o caminho seja o menor possível (MinSum em p^3).

2.3.2 Uma rede de sensores

Considere um sistema de comunicação utilizando uma rede de sensores. Deseja-se decidir qual rota será usada para fazer uma comunicação de um nó sensor r até outro nó sensor t. A existência do arco (i, j) indica que é possível fazer uma comunicação direta do nó sensor i para o nó sensor j. Seja $\hat{p}^1(i)$ a quantidade de energia disponível no nó i e $\bar{p}^1(i, j)$ o gasto de energia para fazer uma transmissão no arco (i, j). Cada arco (i, j) contém três pesos associados.

- $p^1(i, j) = \hat{p}^1(i) - \bar{p}^1(i, j)$: representa a energia remanescente no nó i após um roteamento que utiliza o arco (i, j);
- $p^2(i, j)$: indica o tempo médio de espera em i (tempo que decorre entre a chegada e a saída da informação no nó; o tempo médio de espera dá uma medida de congestionamento no nó);
- $p^3(i, j)$: o custo gasto na transmissão de i para j.

Uma companhia de comunicação que utiliza esta rede pode estar interessada em encontrar um rota de r a t que, simultaneamente:

- Maximize a menor carga remanescente ao longo da rota (MaxMin em p^1);
- Minimize o tempo máximo de espera ao longo da rota (MinMax em p^2);
- Minimize o custo total para fazer a comunicação (MinSum em p^3).

2.3.3 Uma rede de telecomunicações

Considere a construção de uma rede de fibra ótica para disponibilizar Internet banda larga em edifícios comerciais e residenciais. Tipicamente, estas redes são construídas passando os cabos ao longo das ruas/avenidas. Então, o grafo considerado aqui corresponde ao mapa da região. Os nós do grafo são os clientes, que são os edifícios interessados pelo serviço, os cruzamentos entre as ruas (esquinas), e a central de onde será distribuída a Internet (nó raiz). Os arcos são os segmentos de rua ligando: as esquinas adjacentes; os clientes (incluindo o nó raiz) e as duas esquinas, à direita e à esquerda, mais próximas; os clientes localizados entre duas esquinas adjacentes. Para cada nó representando um cliente temos um retorno financeiro associado, que se refere à Internet disponibilizada. Os demais nós não contêm retorno financeiro associado. Necessariamente, alguns clientes devem ser atendidos.

O custo para criar ou dar manutenção em um arco depende do seu tamanho. Um arco muito grande na solução pode causar dificuldades com respeito ao suporte da rede. Como aplicações em telecomunicações usualmente consideram uma probabilidade de falha na comunicação entre as extremidades inicial e final dos arcos (aqui estamos assumindo a mesma probabilidade para todos os arcos, assim como em [9, 16]), então um nó com um grande número de arcos entre ele e a raiz é mais propício a falhas. Portanto, para a construção desta rede podemos estar interessados em uma árvore de Steiner que, simultaneamente:

- Maximize o retorno financeiro;
- Mininize o arco de maior distância;
- Minimize o máximo número de arcos entre a raiz e qualquer nó.

Capítulo 3

Algoritmos para problemas de caminho tri-objetivo

Neste capítulo vamos considerar o caso tri-objetivo do problema (P), definido no capítulo anterior, ou seja, vamos considerar (P) para $l = 2$. Além disso, aqui consideramos o conjunto viável S como sendo o conjunto de todos os caminhos entre um par de nós dado. Cinco algoritmos polinomiais foram desenvolvidos para este problema, onde o objetivo de todos eles é a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas.

Quatro destes algoritmos, MMS, MMS-R, MMS-S e MMS-B trabalham com o fato de que fixando valores assumidos por $p^k(.)$, $k = 1, 2$, podemos gerar grafos parciais do grafo G, isto é, os grafos G_v definidos no capítulo anterior. Como o interesse é um conjunto mínimo completo, então estamos interessados em, no máximo, uma solução Pareto-ótima em cada G_v, que seria a melhor solução neste grafo com relação a função objetivo MinSum. Desta forma, em cada iteração destes algoritmos aplicamos, em um grafo parcial de G, um algoritmo mono-objetivo para o problema MinSum. Aqui, o algoritmo mono-objetivo utilizado foi o algoritmo de Dijkstra implementado com uma heap binária. A complexidade destes quatro algoritmos é a mesma.

O primeiro método desenvolvido para este problema de caminho tri-objetivo foi

o MMS. Para todo $v \in \mathbb{Z}^2$ tal que $1 \leq v_k \leq m_k$, $k = 1, 2$, é resolvido o problema de caminho mínimo em G_v considerando os pesos $p^3(.)$, ou seja, o número de iterações executadas por este algoritmo é sempre $m_1 m_2$. O MMS começa com a iteração $v = (m_1, m_2)$ e segue diminuindo estas coordenadas de v de uma unidade ao longo das iterações, o que implica em um procedimento de inserção de arcos. Um teste de dominância é utilizado pelo algoritmo para garantir que apenas soluções Pareto-ótimas sejam inseridas no conjunto de soluções. Além disso, esse teste garante a não inserção de Pareto-ótimos equivalentes, isto é, soluções com o mesmo vetor objetivo. O MMS é apresentado em [27, 28] (veja o Anexo 1 para mais detalhes sobre este método).

Para os grafos G_v onde v tem coordenadas v_k próximas de m_k, $k = 1, 2$, temos uma grande possibilidade de não existência de soluções viáveis, pois nestes casos temos fortes restrições para o conjunto de arcos M_v. Porém, mesmo com a inexistência de soluções viáveis, o MMS executa tais iterações porque não consegue identificá-las a priori. Para evitar a execução dessas iterações um novo algorimo, MMS-R, foi desenvolvido utilizando uma idéia reversa ao MMS, ou seja, ele trabalha deletando arcos ao longo das iterações. Desta forma, grafos de iterações posteriores a uma iteração v são grafos parciais do grafo da iteração v. Portanto, se for identificada a inexistência de solução viável em uma dada iteração, então, as iterações seguintes não precisam ser executadas. O MMS-R evita essas iterações através de uma variável de controle. Uma matriz de dimensão $m_1 \times m_2$ é utilizada para armazenar as soluções obtidas. Se uma solução s é encontrada em uma iteração $v = (v_1, v_2)$ do algoritmo, então ela é armazenada na posição v dessa matriz de soluções. Além disso, s só será comparada com soluções geradas posteriormente, nas iterações $(v_1, v_2 + 1)$ e $(v_1 + 1, v_2)$. O MMS-R é apresentado em [29, 30]. No Anexo 4 temos a generalização deste algoritmo para o caso multi-objetivo.

Com o intuito de reduzir ainda mais o número de iterações executadas pelo MMS-R, dois novos métodos foram desenvolvidos, MMS-S e MMS-B, os quais trabalham em duas fases. Na primeira fase as soluções são obtidas e inseridas em certas posições da matriz de soluções. A segunda fase é responsável pela eliminação das soluções dominadas e equivalentes. Para atingir essa redução no número de iterações, esses

métodos consideram os valores objetivo de gargalo das soluções obtidas. Estes valores são usados na primeira fase dos algoritmos a fim de armazenar as soluções em posições posteriores às iterações onde elas são obtidas e, com isso, não executar as iterações intermediárias. Ou seja, se em uma dada iteração v obtemos uma solução s, então ela será inserida em uma posição $\bar{v}$ da matriz de soluções tal que $\bar{v} \geq v$, e as iterações $\hat{v}$ tais que $v \leq \hat{v} \leq \bar{v}$ não serão executadas.

No MMS-S, duas variáveis, uma associado a cada objetivo de gargalo, são usadas para indicar quais iterações serão executadas. Elas são incrementadas de acordo com os valores objetivo da solução obtida. O valor dessas variáveis após o incremento indica qual a posição, na matriz de soluções, em que essa solução será inserida. Para evitar que seja pulada alguma iteração que necessita ser executada, o incremento das variáveis nem sempre pode usar exatamente os valores objetivo, o que neste caso faz com que o MMS-S execute mais iterações do que o MMS-B que será apresentado a seguir. Para mais detalhes sobre o MMS-S veja [33].

Para determinar quais as iterações que serão executadas, o algoritmo MMS-B utiliza uma matriz binária R de dimensão $m_1 \times m_2$. Uma iteração $v = (v_1, v_2)$ só será executada se tivermos $R_v = 1$, caso contrário passamos para a próxima iteração, que é $(v_1, v_2 + 1)$ ou $(v_1 + 1, 1)$ se $v_2 = m_2$. Caso uma solução s seja encontrada em uma iteração v do MMS-B, então ela é inserida na posição $\bar{v}$ da matriz de soluções, onde $q_k^s = p_{\bar{v}_k}^k$, $k = 1, 2$. Além disso, fazemos $R_{\hat{v}} = 0$ para todo $\hat{v}$ tal que $v \leq \hat{v} \leq \bar{v}$. Esse vetor $\bar{v}$ é denominado *posição ajustada* da solução s. Veja o Anexo 2 para mais detalhes sobre o algoritmo MMS-B.

O quinto método desenvolvido para o problema, MMS-LS, é um algoritmo de rotulação e tem complexidade pior do que os demais. Este método pode ser visto como uma extensão tri-objetivo do algoritmo de Dijkstra. Ele encontra um conjunto mínimo completo para o problema trabalhando diretamente no grafo original G. Veja o Anexo 2 para mais detalhes.

Capítulo 4

Um algoritmo para um problema de Steiner

Neste capítulo vamos considerar o problema tri-objetivo de árvore de Steiner (P_1), definido no Capítulo 2. Aqui, S é o conjunto de todas as árvores de Steiner no grafo G. Um algoritmo polinomial para a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas foi desenvolvido para o problema.

Considere $d_1, d_2, \ldots, d_{\bar{m}}$ os diferentes valores assumidos pela função $d(i,j)$, $(i,j) \in M$. Logo o número de valores distintos assumidos por $d(i,j)$ é $\bar{m} \leq m$. Sem perda de generalidade, suponha que esses valores estejam em ordem decrescente, ou seja, $d_1 > d_2 > \ldots > d_{\bar{m}}$.

Em cada iteração do algoritmo principal, isto é, do algoritmo para o problema (P_1), se aplica um algoritmo auxiliar (mono-objetivo) no grafo parcial de G definido por $G_k = (N, M_k)$, onde $M_k = \{(i,j) \in M \mid d(i,j) \leq d_k\}$. Este algoritmo auxiliar tenta encontrar uma árvore de Steiner ótima, com relação à terceira função objetivo de (P_1), e com no máximo h arcos intermediários entre a raiz e qualquer nó desta árvore. As variáveis, k e h, são definidas em cada iteração do algoritmo principal e passadas para o algoritmo auxiliar.

Um procedimento similar a esse algoritmo auxiliar foi desenvolvido por Gabow e Tarjan [14] para encontrar uma árvore geradora de G, ao invés de uma árvore de

Steiner. Esses dois algoritmos fazem uma busca em largura em um grafo parcial do grafo original.

O algoritmo principal é similar ao algoritmo MMS-B, apresentado no capítulo anterior para um probema de caminho tri-objetivo (veja o Anexo 3 para mais detalhes).

Capítulo 5

Algoritmos multi-objetivo

Neste capítulo vamos considerar o problema multi-objetivo (P), definido no Capítulo 2. Ou seja, aqui o número de funções objetivo de gargalo, l, é um inteiro positivo qualquer dado. Dois dos algoritmos apresentados no Capítulo 3 foram generalizados para esse problema multi-objetivo, o MMS-R e o MMS-B. A generalização do MMS-R ocorreu antes do surgimento do MMS-B. Como este último é mais eficiente, então não faria muito sentido generalizar os dois algoritmos ao mesmo tempo, ou seja, bastaria a generalização do MMS-B.

Esses dois algoritmos multi-objetivo utilizam uma matriz $\bar{S}$ para armazenar as soluções geradas e outra, $\bar{Q}$, a qual nos permite determinar os vetores objetivo das soluções em $\bar{S}$. Ambas matrizes são de dimensão $m_1 \times m_2 \times \cdots \times m_l$. Se no final dos algoritmos tivermos uma solução s na posição v de $\bar{S}$, então nesta mesma posição de $\bar{Q}$ temos o valor real q^s_{l+1}. Para as l primeiras coordenadas do vetor objetivo q^s temos que $q^s_k = p^k_{v_k}$, $k = 1, 2, \ldots, l$.

No MMS-R, além das duas matrizes, são usadas l variáveis de controle, $ctrl_1, ctrl_2, \ldots, ctrl_l$, para evitar a execução das iterações desnecessárias, que são aquelas para as quais sabemos que não existem soluções viáveis. Veja o Anexo 4 para mais detalhes sobre este algoritmo.

5.1 Algoritmo MMS-B generalizado

As l variáveis de controle utilizadas no MMS-R dificultam um pouco o entendimento do algoritmo. Além disso, a matriz binária R, usada no algoritmo MMS-B para indicar quais iterações serão executadas, pode também ser usada para evitar estas iterações para as quais sabemos da inexistência de soluções viáveis. Então, para simplificar o entendimento da versão multi-objetivo do MMS-B, aqui vamos descrever este algoritmo sem as variáveis de controle. Porém, estas variáveis podem ser introduzidas no algoritmo da mesma forma que são usadas no MMS-R. Aqui, a matriz R é de dimensão $m_1 \times m_2 \times \cdots \times m_l$, ao invés de $m_1 \times m_2$.

Segue-se o algoritmo MMS-B generalizado para o caso multi-objetivo. No algoritmo, o vetor $e_j \in \mathbb{Z}^l$ é um vetor de zeros com 1 na posição j. *AlgS*, que também aparece no MMS-B, é um algoritmo mono-objetivo para o problema MinSum. Ele é aplicado, em cada iteração v do MMS-B, no grafo G_v considerando os pesos $p^{l+1}(.)$. Portanto, na primeira fase do MMS-B as soluções são obtidas e inseridas em $\bar{S}$. Nesta fase a matriz R controla quais iterações serão executadas, ou seja, ela indica em quais grafos G_v deve se aplicar *AlgS*. Na segunda fase a matriz $\bar{Q}$ é usada para comparar as solução de $\bar{S}$ e, assim, eliminar as soluções dominadas e equivalentes.

Algoritmo 5.1.1 *MMS-B multi-objetivo*

1. Faça: $\bar{S}_v \leftarrow \emptyset$; $\bar{Q}_v \leftarrow \infty$; $R_v \leftarrow 1$, $\forall v$, $1 \leq v_k \leq m_k$, $k = 1, 2, \ldots, l$

Fase 1. Encontrando soluções candidatas a Pareto-ótimas

2. Para v_1 De 1 Até m_1 Faça

2.1. Para v_2 De 1 Até m_2 Faça

$\vdots$

2.1.1. Para v_l De 1 Até m_l Faça

Se $R_v = 1$ Então

Aplique *AlgS* em G_v considerando $p^{l+1}(.)$

Se não existe solução em G_v Então $R_{\bar{v}} \leftarrow 0$, $\forall \bar{v} \geq v$

Senão, seja s a solução obtida por *AlgS* com $q^s_k = p^k_{\hat{v}_k}$, $k = 1, 2, \ldots, l$

Se $\bar{S}_{\hat{v}} = \emptyset$ Então $\bar{S}_{\hat{v}} \leftarrow s$; $\bar{Q}_{\hat{v}} \leftarrow q^s_{l+1}$

$R_{\bar{v}} \leftarrow 0$, $\forall \bar{v}$ tal que $v \leq \bar{v} \leq \hat{v}$

Fase 2. Deletando soluções dominadas e equivalentes

3. Para v_1 De m_1 Até 1 Faça

3.1. Para v_2 De m_2 Até 1 Faça

$\vdots$

3.1.1. Para v_l De m_l Até 1 Faça

Para k De 1 Até l Faça

Se $v_k > 1$ e $\bar{Q}_v \leq \bar{Q}_{v-e_k}$ Então $\bar{S}_{v-e_k} \leftarrow \emptyset$; $\bar{Q}_{v-e_k} \leftarrow \bar{Q}_v$

A complexidade do algoritmo MMS-B para o problema multi-objetivo (P) é $O(m^l p(n))$, onde $p(n)$ é a complexidade de $AlgS$. Temos que a complexidade do laço mais interno da segunda fase é $O(1)$, pois estamos supondo $l \leq \lambda$, onde λ é uma constante.

Na primeira fase soluções são obtidas e inseridas em $\bar{S}$. Suponhamos que para uma determinada iteração $v = (v_k)$ foi obtida uma solução s tal que $q^s_k = p^k_{\hat{v}_k}$ para $k = 1, 2, \ldots, l$. Caso $\hat{v} = v$ então $q^s_k = p^k_{v_k}$ para $k = 1, 2, \ldots, l$, o que significa que s não tem 'folga', isto é, ela explora ao máximo os limites do grafo G_v. Neste caso nenhum 'salto' de iterações será possível.

Suponhamos agora uma outra situação em que $\hat{v}_k = v_k$ para $k = 1, 2, \ldots, g-1$, $g+1, \ldots, l$ e $\hat{v}_g = v_{g+2}$. Temos então $q^s_k = p^k_{v_k}$ para $k = 1, 2, \ldots, g-1, g+1, \ldots, l$ e $q^s_g = p^g_{v_{g+2}}$. Logo nas iterações $\bar{v}$ e $\tilde{v}$ onde $\bar{v}_k = \tilde{v}_k = v_k$ para $k = 1, 2, \ldots, g-1$, $g+1, \ldots, l$, $\bar{v}_g = v_{g+1}$ e $\tilde{v}_g = v_{g+2}$, os grafos gerados $G_{\bar{v}}$ e $G_{\tilde{v}}$ são grafos parciais de G_v, o que significa que na minimização da função soma feita por $AlgS$ nenhum valor melhor que q^s_{l+1} poderá ser obtida. Mas pelo fato dos pesos dos arcos da solução s não serem maiores do que $p^k_{\bar{v}_k}$ e $p^k_{\tilde{v}_k}$ então s é uma solução viável em $G_{\bar{v}}$ e $G_{\tilde{v}}$. Portanto s é uma solução ótima com relação ao objetivo MinSum nos grafos $G_{\bar{v}}$ e $G_{\tilde{v}}$. Este fato possibilita o 'salto' das iterações $\bar{v}$ e $\tilde{v}$, acelerando a convergência do algoritmo. O teste $\bar{S}_{\hat{v}} = \emptyset$ visa meramente evitar que soluções com os mesmos valores para as funções objetivo sejam sobrepostas. Caso $\bar{S}_{\hat{v}} \neq \emptyset$ então a solução armazenada em $\bar{S}_{\hat{v}}$ tem o vetor objetivo igual a q^s, o que torna desnecessário armazenar s. Todos os demais casos em que é obtida uma solução s na iteração v são extensões do caso ilustrado.

Para terminar a apresentação da primeira fase, consideremos o caso de não existir solução viável em G_v na iteração v. Ora, se G_v não tem solução viável, então evidentemente também não existe solução viável nos grafos $G_{\bar{v}}$ para $\bar{v} \geq v$, pois estes grafos são grafos parciais de G_v (veja a seção 2.1). Desta forma podemos 'pular' estas iterações $\bar{v} \geq v$. Isto é feito no algoritmo fazendo $R_{\bar{v}} \leftarrow 0$.

Denominemos $\bar{S}_1^*$ o conjunto formado pelas soluções contidas na matriz $\bar{S}$ no final da primeira fase. O Teorema 5.1.2 mostra que $\bar{S}_1^*$ contém um conjunto mínimo completo de soluções Pareto-ótimas S^*, ou seja, para qualquer Pareto-ótimo $s^* \in S$ existe uma solução equivalente $s \in \bar{S}_1^*$, isto é, $q^s = q^{s^*}$.

A segunda fase do Algoritmo 5.1.1 faz uma depuração de $\bar{S}$, retirando soluções que não são Pareto-ótimas. Seja $\bar{S}_2^*$ o conjunto formado pelas soluções contidas na matriz $\bar{S}$ no final da segunda fase. O Teorema 5.1.3 demonstra que $\bar{S}_2^* = S^*$ através dos seguintes fatos:

1. Não existem duas soluções em $\bar{S}_1^*$ com o mesmo vetor objetivo. Como na segunda fase somente se elimina soluções isto também se aplica a $\bar{S}_2^*$.

2. Qualquer solução $s \in \bar{S}_1^* - S^*$ é dominada por alguma solução $s^* \in S^*$ sendo portanto eliminada na segunda fase.

3. Nenhuma solução $s^* \in S^*$ é eliminada na segunda fase.

Os resultados a seguir garantem a otimalidade do algoritmo, isto é, que ele de fato gera um conjunto mínimo completo de soluções Pareto-ótimas para o problema (P). Quando dissermos apenas iteração v, estamos nos referindo à primeira fase.

Teorema 5.1.2 *Seja s^* um Pareto-ótimo qualquer e seja $\bar{v} = (\bar{v}_k)$ tal que $q_k^{s^*} = p_{\bar{v}_k}^k$, $k = 1, 2, \ldots, l$. Então no final da primeira fase do Algoritmo 5.1.1 temos $\bar{S}_{\bar{v}} = s$ onde s é tal que $q^s = q^{s^*}$.*

Demonstração: Como s^* é solução de $G_{\bar{v}}$, ou seja, existe solução em $G_{\bar{v}}$, então no final da primeira fase, em princípio, existem as seguintes possibilidades no que diz respeito à posição $\bar{v}$ de $\bar{S}$:

(a) $\bar{S}_{\bar{v}} = \emptyset$. Isto implica que em uma iteração v *AlgS* determinou uma solução s onde $q_k^s = p_{\hat{v}_k}^k$ com $v \leq \bar{v} \leq \hat{v}$ e $\bar{v} \neq \hat{v}$.

(b) $\bar{S}_{\bar{v}} = s$. Isto implica que em uma iteração v *AlgS* determinou uma solução s onde $q_k^s = p_{\hat{v}_k}^k$ com $v \leq \bar{v} = \hat{v}$.

Em ambos os casos podemos garantir que existe uma solução s em $\bar{S}$, gerada por *AlgS* na iteração v, tal que $q_k^s = p_{\hat{v}_k}^k \leq p_{\bar{v}_k}^k = q_k^{s^*}$ para $k = 1, 2, \ldots, l$. Estas l desigualdades são garantidas pelo fato de $\bar{v} \leq \hat{v}$ (veja (2.1)). O fato de s^* estar em $G_{\bar{v}}$ garante que s^* também está em G_v pois $G_{\bar{v}}$ é grafo parcial de G_v. A otimalidade de *AlgS* na iteração v garante $q_{l+1}^s \leq q_{l+1}^{s^*}$ o que resulta em $q^s \leq q^{s^*}$. O fato de s^* ser Pareto-ótimo garante necessariamente que $q^s = q^{s^*}$. Por esta última igualdade temos $p_{\hat{v}_k}^k = p_{\bar{v}_k}^k$, $k = 1, 2, \ldots, l$, implicando em $\hat{v} = \bar{v}$, o que por sua vez impossibilita o caso (a). Portanto no final da primeira fase temos $\bar{S}_{\bar{v}} = s$. ∎

Teorema 5.1.3 *No final da segunda fase do Algoritmo 5.1.1 as soluções em $\bar{S}$ formam um conjunto mínimo completo.*

Demonstração: O Teorema 5.1.2 garante que o conjunto de soluções em $\bar{S}$ no final da primeira fase, $\bar{S}_1^*$, contém um conjunto mínimo completo S^*. Se $s \in \bar{S}_1^* - S^*$ duas situações se apresentam:

(a) Existe $s^* \in S^*$ tal que $q^s = q^{s^*}$. Mas isto é impossível pois s e s^* estão na matriz $\bar{S}$. Ora, nesta matriz as soluções estão em posições distintas significando portanto que $q^s \neq q^{s^*}$. (Basta ver que as l primeiras componentes de q^s e q^{s^*} são determinadas pela posição ocupada em $\bar{S}$).

(b) Existe $s^* \in S^*$ tal que $q^{s^*} \leq q^s$ com $q^{s^*} \neq q^s$, ou seja, s é dominada por s^*. Seja $q_k^s = p_{v_k}^k$ e $q_k^{s^*} = p_{v_k^*}^k$ para $k = 1, 2, \ldots, l$. O fato de termos $q_k^{s^*} \leq q_k^s$ nos garante que $v_k^* \geq v_k$. Logo $v^* \geq v$, onde $v = (v_k)$ e $v^* = (v_k^*)$. Temos que v e v^* são, respectivamente, as posições de s e s^* em $\bar{S}$. Como temos no máximo uma solução em cada posição de $\bar{S}$, então $v^* \geq v$ com $v^* \neq v$.

Por outro lado temos $q_{l+1}^{s^*} \leq q_{l+1}^s$ o que implica em $\bar{Q}_{v^*} \leq Q_v$. Por transitividade, podemos verificar que o Algoritmo 5.1.1 durante a segunda fase, para $\bar{Q}_{v^*} \leq Q_v$ e $v \leq v^*$ com $v \neq v^*$ faz $\bar{S}_v \leftarrow \emptyset$. Ou seja, uma solução s que é dominada por uma solução Pareto-ótima, é eliminada de $\bar{S}$ até o final da segunda fase.

Para terminar esta demonstração basta mostrar que nenhuma solução Pareto-ótima é eliminada na segunda fase. Seja $s^* \in \bar{S}_1^*$ uma solução Pareto-ótima. Então no final da primeira fase temos $\bar{S}_{v^*} = s^*$, onde $q_k^{s^*} = p_{v_k^*}^k$. Pela segunda fase a única possibilidade de s^* ser eliminada ($\bar{S}_{v^*} \leftarrow \emptyset$) é existir $\bar{S}_v = s$ com $v \geq v^*$ e $v \neq v^*$ tal que $\bar{Q}_v = q_{l+1}^s \leq \bar{Q}_{v^*} = q_{l+1}^{s^*}$. O fato de termos $v \geq v^*$ com $v \neq v^*$ garante $q_k^s = p_{v_k}^k \leq p_{v_k^*}^k = q_k^{s^*}$ para $k = 1, 2, \ldots, l$ com pelo menos uma desigualdade estrita.

Estas l desigualdades junto com $q^s_{l+1} \leq q^{s^*}_{l+1}$ implicam em $q^s \leq q^{s^*}$ com $q^s \neq q^{s^*}$, o que contradiz a suposição de que s^* é Pareto-ótimo.
Portanto podemos garantir que no final da segunda fase temos $\bar{S}^*_2 = S^*$.

∎

O Algoritmo 5.1.1 foi implementado para os problemas de caminho e árvore geradora. Para caminho consideramos o problema (P), definido no Capítulo 2, com $l = 2$ e 3, e para árvore geradora consideramos apenas o caso tri-objetivo de (P), isto é, $l = 2$. Os resultados destas implementações são apresentados a seguir.

5.2 Resultados computacionais

Nesta seção vamos apresentar resultados computacionais referentes às implementações dos algoritmos para a obtenção de um conjunto mínimo completo. A grande maioria dos experimentos realizados se refere ao problema de caminho tri-objetivo MinMax-MinMax-MinSum. Em Pinto [25] e no Anexo 1, experimentos foram feitos para o algoritmo MMS e, em Pinto *et al* [29], os experimentos comparam o MMS-R com o MMS. No Anexo 2, os sete algoritmos descritos no Capítulo 3 foram comparados. Em todas as instâncias resolvidas para comparar estes sete métodos, o algoritmo MMS-B foi o mais eficiente. Por este motivo, para os novos experimentos apresentados aqui consideramos apenas o MMS-B. No Anexo 3 temos resultados computacionais considerando o algoritmo do Capítulo 4, desenvolvido para o problema tri-objetivo de árvore de Steiner apresentado no Capítulo 2.

Considerando o problema (P) com $l = 2$, foi feita uma implementação do algoitmo MMS-B para o problema tri-objetivo da árvore geradora em grafos não-orientados. Nesta implementação, o algoritmo de Prim [35] utilizando uma *heap* binária foi considerado para *AlgS*. Além disso, considerando (P) com $l = 3$, uma implementação do MMS-B para o problema de caminho com quatro objetivos em grafos orientados, também foi desenvolvida. Neste problema de caminho, para *AlgS* foi feita uma implementação do algoritmo de Dijkstra [11] com uma *heap* binária. As implementações foram desenvolvidas em linguagem C e executadas em um PC

com um processador Intel Core 2 Duo de $2.0\,GHz$ e memória RAM de $3\,Gb$.

Duas tabelas (uma para cada problema citado acima) são apresentadas, onde cada linha representa a média de dez instâncias. As colunas *iterações* e #PO destas tabelas representam, respectivamente, o número de vezes em que o *AlgS* foi chamado pelo algoritmo MMS-B e o número de soluções Pareto-ótimas em $\bar{S}$ no final da execução do MMS-B, isto é, #PO é a cardinalidade do conjunto mínimo completo.

Grafos aleatórios não-orientados com n nós, $n = 500, 2\,000, 7\,000$ e $10\,000$, foram gerados para este problema tri-objetivo da árvore geradora. Para cada nó, d nós adjacentes são gerados aleatoriamente, $d = 8, 10$. Uma verificação é feita durante a geração destes arcos garantindo que o grau de cada nó seja no máximo d. Desta forma, pela não-orientação destes grafos temos que o número de arcos m será sempre menor ou igual a $dn/2$, como pode ser visto na Tabela 5.1. Os três pesos de cada arco, $p^1(.), p^2(.)$ e $p^3(.)$, também são gerados aleatoriamente. Para $p^1(.)$ e $p^2(.)$, que estão associados as funções gargalo, geramos inteiros nos intervalos $[1, 50]$ e $[1, 500]$. Para os pesos associados a função objetivo MinSum, $p^3(.)$, foram gerados inteiros no intervalo $[1, 1\,000]$.

Um importante comentário sobre a Tabela 5.1 se refere a cardinalidade do conjunto mínimo completo (#PO). O limite teórico é dado por $m_1 m_2$, que neste caso é $2\,500$ para $m_k = 50$ e $250\,000$ para $m_k = 500$, $k = 1, 2$. Porém, estes experimentos mostram que #PO pode ser bem menor. Para $m_k = 50$ temos que #PO está entre $0,2\%$ e 15% de $m_1 m_2$ e, para $m_k = 500$, #PO está entre $0,1\%$ e $6,7\%$ de $m_1 m_2$.

Uma outra importante observação diz respeito à diferença entre #PO e o número de vezes que *AlgS* é executado (coluna *iterações*). Esta diferença foi muito pequena, o que é um resultado muito bom para o algoritmo MMS-B, pois como o mecanismo destes algoritmos consiste em encontrar no máximo uma solução por iteração, então temos que #PO é um limite inferior para o número de iterações.

Para o problema de caminho com quatro objetivos citado no início desta seção, foram gerados grafos aleatórios orientados com n nós, $n = 1\,000, 5\,000, 8\,000$ e $15\,000$. Para cada nó, d sucessores são escolhidos aleatoriamente, $d = 5, 20$, o que implica

n	m	$m_1 = m_2$	#PO	iterações	segundos
500	1 907	50	179	183	0.06
500	1 906	500	7 282	7 300	2.56
500	2 408	50	383	389	0.14
500	2 404	500	16 763	16 800	6.19
2 000	7 621	50	66	69	0.12
2 000	7 617	500	5 329	5 332	9.01
2 000	9 621	50	180	184	0.35
2 000	9 606	500	16 548	16 553	29.48
7 000	26 421	50	5	7	0.05
7 000	26 411	500	257	258	1.86
7 000	33 286	50	43	45	0.36
7 000	33 306	500	5 836	5 839	42.43
10 000	37 850	50	7	8	0.10
10 000	37 847	500	701	703	7.19
10 000	47 774	50	58	61	0.74
10 000	47 774	500	6 429	6 433	71.03

Tabela 5.1: *MMS-B para o problema tri-objetivo da árvore geradora* (P).

em $m = dn$ arcos. Para os três pesos associados às funções gargalo, $p^1(.), p^2(.)$ e $p^3(.)$, são gerados inteiros aleatórios nos intervalos $[1, 50]$ e $[1, 100]$. Para os pesos associados à função objetivo MinSum, $p^4(.)$, foram gerados inteiros no intervalo $[1, 1\,000]$.

Na Tabela 5.2 a diferença entre #PO e seu limite teórico, que aqui é $m_1 m_2 m_3$, é ainda maior do que na Tabela 5.1. #PO ficou entre 0.3% e 2% de $m_1 m_2 m_3$, para $m_1 = m_2 = m_3 = 50$, e entre $0,04\%$ e $0,5\%$ de $m_1 m_2 m_3$, para $m_1 = m_2 = m_3 = 100$. Por outro lado, a diferença entre #PO e o número de vezes que *AlgS* é executado cresceu. Porém, esta diferença continua pequena. Aqui o número de execuções do *AlgS* foi sempre menor do que $2,6$ #PO.

n	m	$m_1 = m_2 = m_3$	#PO	iterações	segundos
1 000	5 000	50	332	711	0.17
1 000	5 000	100	420	1 075	0.28
1 000	20 000	50	1 294	2 605	1.44
1 000	20 000	100	1 939	4 736	2.53
5 000	25 000	50	522	998	1.53
5 000	25 000	100	931	2 071	2.88
5 000	100 000	50	2 025	3 810	13.42
5 000	100 000	100	2 839	6 213	21.70
8 000	40 000	50	722	1 348	3.25
8 000	40 000	100	978	2 134	4.87
8 000	160 000	50	2 441	4 552	27.18
8 000	160 000	100	4 184	9 113	53.40
15 000	75 000	50	765	1 370	7.65
15 000	75 000	100	926	2 007	10.65
15 000	300 000	50	2 448	4 487	56.91
15 000	300 000	100	4 712	10 133	112.55

Tabela 5.2: *MMS-B para o problema de caminho com quatro objetivos* (P).

Capítulo 6

Considerações finais

Este trabalho resultou no desenvolvimento de algoritmos polinomiais para a obtenção de um conjunto mínimo completo de soluções Pareto-ótimas para problemas multi-objetivo em grafos. Várias implementações para o problema de caminho tri-objetivo com duas funções objetivo MinMax e uma MinSum foram desenvolvidas e comparadas. Além disso, também foram desenvolvidas implementações para o problema de caminho com quatro objetivos e para os problemas tri-objetivo de árvore geradora e de árvore de Steiner. Em todos estes problemas temos apenas uma função objetivo totalizadora. Todas as demais funções objetivo consideradas são de gargalo.

Os problemas considerados nesta tese ainda não haviam sido tratados na literatura de otimização combinatória multi-objetivo. Por este motivo, aplicações foram apresentadas para justificar o nosso interesse pelos problemas. Os trabalhos anteriores mais próximos são os de Hansen [18], Martins [23] e Berman *et al* [4], onde algoritmos bi-objetivo são apresentados para o problema de caminho com uma função MinSum e uma MinMax.

Os experimentos computacionais realizados mostraram que o algoritmo MMS-B foi sempre mais eficiente do que os demais métodos, mesmo tendo a mesma complexidade no pior caso. Isto se deve ao fato de que, teoricamente, o número de vezes que *AlgS* é executado no MMS-B é sempre menor ou igual comparado aos outros

métodos, e para os problemas gerados nos testes computacionais, a quantidade de execuções de *AlgS* pelo MMS-B foi sempre estritamente menor.

No problema de árvore de Steiner considerado neste trabalho, em uma das funções objetivo de gargalo, ao invés de minimizar o máximo (ou maximizar o mínimo) peso dos arcos, para algum peso $p^k(.)$, o interesse consiste em minimizar o número de nós intermediários entre a raiz e o nó mais distante. Isto se deve à aplicação que motivou o estudo deste problema tri-objetivo, apresentada no Capítulo 2. Mesmo considerando este tipo de função gargalo foi possível desenvolver um algoritmo polinomial para o problema, o qual é similar ao algoritmo MMS-B. Portanto, assim como foi possível tratar este problema tri-objetivo de árvore de Steiner em tempo polinomial, possivelmente outros problemas de otimização combinatória multi-objetivo também podem ser resolvidos polinomialmente adaptando o algoritmo MMS-B.

Antes de trabalhar na adaptação do algoritmo MMS-B para outros problemas de otimização combinatória, julgamos que seja importante apresentar aplicações para justificar o interesse pelos problemas. Da mesma forma que na aplicação para o problema tri-objetivo de árvore de Steiner surgiu uma função gargalo diferente, isto é, definida sobre os nós e não sobre os arcos, é possível que as funções objetivo para esses outros problemas também tenham características particulares. Logo, a adaptação do algoritmo vai depender das aplicações.

Restrições adicionais podem ser consideradas pelos problemas que apresentamos no Capítulo 2. Mesmo incorporando restrições de gargalo aos problemas (P) e (P_1), ou seja:

$$\max_{(i,j)\in s}\{p^k(i,j)\} \leq \beta \quad \text{ou} \quad \min_{(i,j)\in s}\{p^k(i,j)\} \geq \beta,$$

onde β é um valor dado e $p^k(.)$ é um peso adicional (ou pode ser algum dos pesos já existentes), eles continuam podendo ser resolvidos polinomialmente pelos algoritmos apresentados nesta tese. Para este tipo de restrições basta eliminar, antes da execução dos algoritmos, os arcos (i,j) tais que $p^k(i,j) > \beta$ ou $p^k(i,j) < \beta$, respectivamente. Esta eliminação de arcos implica na redução de pesos distintos assumidos pelas funções gargalo. Temos que essa quantidade de pesos distintos limita

o número de iterações executadas pelos algoritmos. Logo, os algoritmos ficam mais eficientes caso restrições de gargalo sejam incorporadas aos problemas.

Caso restrições do tipo totalizadoras sejam incorporadas ao problema (P), como por exemplo:

$$\sum_{(i,j)\in s} p^k(i,j) \leq \beta \quad \text{ou} \quad \prod_{(i,j)\in s} p^k(i,j) \geq \beta,$$

com $k \neq l+1$, então (P) se torna NP-difícil, por exemplo, para o problema de caminho (veja Hansen [18]) e para o problema da árvore geradora (veja Camerini *et al* [5]).

O acréscimo de uma restrição totalizadora ao problema (P_1), como a primeira do parágrafo acima, leva a uma variante do *Prize-Collecting Steiner Tree Problem* que é NP-difícil (veja Costa *et al* [9]).

Referências Bibliográficas

[1] Ahuja, R. K., Magnanti, T. L., Orlin, J. B. "Network Flows: Theory, Algorithms and Applications". Prentice Hall, New Jersey, 1993.

[2] Azevedo, J. A., Martins, E. Q. V. "An algorithm for the multiobjective shortest path problem on acyclic networks". *Investigação Operacional*, v. 11 (1), pp. 52–69, 1991.

[3] Bellman, R. E. "On a routing problem". *Quart. Appl. Math.*, v. 16, pp. 87–90, 1958.

[4] Berman, O., Einav, D., Handler, G. "The constrained bottleneck problem in network". *Operations Research*, v. 38, pp. 178–181, 1990.

[5] Camerini, P. M., Galbiati, G., Maffioli, F. "The complexity of multi-constrained spanning tree problems". In: *Theory of Algorithms*, L. Lovasz (ed.), Colloquium, Pecs 1984. North-Holland, Amsterdam 53–101, 1984.

[6] Canuto, S. A., Resende, M. G. C., Ribeiro, C. C. "Local search with perturbations for the prize-collecting Steiner tree problem in graphs". *Networks*, v. 38, pp. 50–58, 2001.

[7] Clímaco, J. C. N., Martins, E. Q. V. "A bicriterion shortest path algorithm". *European Journal of Operational Research*, v. 11, pp. 399–404, 1982.

[8] Costa, A. M., Cordeau, J. -F., Laporte, G. "Steiner tree problems with profits". *INFOR*, v. 44, pp. 99–115, 2006.

[9] Costa, A. M., Cordeau, J. -F., Laporte, G. "Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints". *Networks*, v. 53, pp. 141–159, 2008.

[10] Cunha, A. S., Lucena, A., Maculan, N., Resende, M. G. C. "A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs". *Discrete Applied Mathematics*, v. 157, pp. 1198–1217, 2009.

[11] Dijkstra, E. W. "A note on two problems in connexion with graphs". *Numer. Math.*, v. 1, pp. 269–271, 1959.

[12] Ehrgott M., Gandibleux, X. "A survey and annotated bibliography of multiobjective combinatorial optimization". *OR Spektrum*, v. 22, pp. 425–460, 2000.

[13] Gabow, H. N., Galil, Z., Spencer, T., Tarjan, R. E. "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs". *Combinatorica*, v. 6, pp. 109–122, 1986.

[14] Gabow, H. N., Tarjan, R. E. "Algorithms for two bottleneck optimization problems". *Journal of Algorithms*, v. 9, pp. 411–417, 1988.

[15] Gandibleux, X., Beugnies, F., Randriamasy, S. "Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function". *A Quarterly Journal of Operations Research*, v. 4, pp. 47–59, 2006.

[16] Gouveia, L. "Multicommodity flow models for spanning trees with hop constraints". *European Journal of Operational Research*, v. 95, pp. 178–190, 1996.

[17] Guerriero, F., Musmanno R. "Label correcting methods to solve multicriteria shortest path problems". *Journal of Optimization Theory and Applications*, v. 111, n. 3, pp. 589–613, 2001.

[18] Hansen, P. "Bicriterion path problems". In: *Multicriteria decision making: theory and applications*, Lecture Notes in Economics and Mathematical Systems, 177, G. Fandel and T. Gal (eds.), Springer, Heidelberg, 109–127, 1980.

[19] Johnson, D. S., Minkoff, M., Phillips, S. "The prize collecting Steiner tree problem: theory and practice". *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, 2000.

[20] Kruskal, J. B. "On the Shortest Spanning Subtree of a graph and the Travelling Salesman Problem". *Proc. Amer. Math. Soc.*, v. 7, pp. 48–50, 1956.

[21] Koopmans, T. C. "Analysis of production as an efficient combination of activities". In: *Activity Analysis of Production and Allocation* (Chap. III), ed. T. C. Koopmans, John Wiley & Sons, New York, pp. 33–97, 1951.

[22] Martins, E. Q. V. "On a multicriteria shortest path problem". *European Journal of Operational Research*, v. 16, pp. 236–245, 1984.

[23] Martins, E. Q. V. "On a special class of bicriterion path problems". *European Journal of Operational Research*, v. 17, pp. 85-94, 1984.

[24] Pareto, V. "Course d'Economic Politique". Lausanne, Rouge, 1896.

[25] Pinto, L. L. "Um algoritmo polinomial para problemas tri-objetivo de otimização de caminhos em grafos". 2007. 34 p. Dissertação (Mestrado em Engenharia de Sistemas e Computação) - Programa de Engenharia de Sistemas e Computação, COPPE, UFRJ, Rio de Janeiro–RJ, 2007.

[26] Pinto, L. L., Bornstein, C. T. "Software CamOtim". Disponível em: *http://www.cos.ufrj.br/camotim*, 2006.

[27] Pinto, L. L., Bornstein, C. T. "Efficient solutions for the multi-objective optimal path problem in graphs". *Anais do Première Conference Internationale en Calcul de Variations et Recherche Operationelle*, Ouidah–Benin, 2007.

[28] Pinto, L. L., Bornstein, C. T., Maculan N. "The tricriterion shortest path problem with at least two bottleneck objective functions". *European Journal of Operational Research*, v. 198, pp. 387–391, 2009.

[29] Pinto, L. L., Bornstein, C. T., Maculan N. "Um problema de caminho tri-objetivo". *Anais do XL SBPO*, João Pessoa–PB, pp. 1032–1042, 2008.(http://www.cos.ufrj.br/∼leizer/MMS-R.pdf)

[30] Pinto, L. L., Bornstein, C. T., Maculan N. "A reverse algorithm for the tricriterion shortest path problem". *Anais do XXXIX Annual Conference of Italian Operational Research Society*, Ischia–Italy, 2008.

[31] Pinto, L. L., Bornstein, C. T., Maculan N. "A polynomial algorithm for the multi-objective bottleneck network problem with one MinSum objective function". Submetido para *Networks*, 2008.

[32] Pinto, L. L., Laporte, G. "An effcient algorithm for the Steiner tree problem with revenue, bottleneck and hop objective functions". Submetido para *European Journal of Operational Research*, 2009.

[33] Pinto, L. L., Pascoal, M. "Enhanced algorithms for tricriteria shortest path problems with two bottleneck objective functions". *Relatórios de Investigação*, n. 3, INESC–Coimbra, 2009. (http://www.inescc.pt/documentos/3 2009.pdf)

[34] Pinto, L. L., Pascoal, M. "On algorithms for the tricriteria shortest path problem with two bottleneck objective functions". Submetido para *Computers & Operations Research*, 2009.

[35] Prim, R. C. "Shortest connection networks and some generalisations". *Bell System Technical Journal*, v. 36, pp. 1389–1401, 1957.

[36] Serafini, P. "Some considerations about computational complexity for multi objective combinatorial problems". In: *Recent advances and historical de-*

velopment of vector optimization. Lecture Notes in Economics and Mathematical Systems, 294, W. Krabs and J. Jahn (eds.), Springer, Berlin, Heidelberg, New York, 222–232, 1987.

[37] Tung, C. T., Chew, K. L. "A bicriterion Pareto–optimal path algorithm". *Asia–Pacific Journal of Operational Research*, v. 5, pp. 166–172, 1988.

[38] Tung, C. T., Chew, K. L. "A multicriteria Pareto–optimal path algorithm". *European Journal of Operational Research*, v. 62, pp. 203–209, 1992.

[39] Zeleny, M. "Linear Multiobjective Programming". *Lecture Notes in Economics and Mathematical Systems*, 95, Springer-Verlag, 1974.

[40] Zelikovsky, A. "Bottleneck Steiner Tree Problems". In: *Encyclopedia of Optimization*, 2nd edition, C. A. Floudas and P. M. Pardalos (Eds.), Springer, New York, 311–313, 2009.

Anexo 1

Este anexo contém o artigo:

Pinto, L. L., Bornstein, C. T., Maculan N. "The tricriterion shortest path problem with at least two bottleneck objective functions". *European Journal of Operational Research*, v. 198, pp. 387–391, 2009.

The tricriterion shortest path problem with at least two bottleneck objective functions

Leizer de Lima Pinto
Cláudio Thomás Bornstein
Nelson Maculan

COPPE/UFRJ – Federal University of Rio de Janeiro, Brazil
Dept. of Systems Engineering and Computer Science

Abstract: The focus of this paper is on the tricriterion shortest path problem where two objective functions are of the bottleneck type, for example MinMax or MaxMin. The third objective function may be of the same kind or we may consider, for example, MinSum or MaxProd. Let $p(n)$ be the complexity of a classical single objective algorithm responsible for this third function where *n* is the number of nodes and *m* be the number of arcs of the graph. An $O(m^2 p(n))$ algorithm is presented that can generate the minimal complete set of Pareto-optimal solutions. Finding the maximal complete set is also possible. Optimality proofs are given and extensions for several special cases are presented. Computational experience for a set of randomly generated problems is reported.

Keywords: Multicriteria shortest path problem, Pareto-optimal solution.

1) Introduction:

Traditionally the shortest path (SP) problem considers just one objective function which generally consists in minimizing the sum (MinSum) of the weights of the path. This problem may be solved for non-negative weights by a label-setting (LS) algorithm, Dijkstra for example, or in the general case by a label-correcting (LC) algorithm. For more details see Ahuja *et al.* (1993). Gondran & Minoux (1986) generalize these results considering several other functions like MaxSum, MaxProd and MaxMin. An extension for the MinMin and MaxMax problem is also possible but few applications for these cases have been reported. Additionally, the results may be extended for the MinProd and MinMax cases. A free software for the optimization of the problems mentioned above is available at Pinto & Bornstein (2006a) and is reported in Pinto & Bornstein (2006b).

Many SP-problems consider more than one objective function. Applications for the multi-objective SP-problem are given in Batta & Chiu (1988), Current & Min (1986) and Current & Marsh (1993). For the multi-objective problem, also called the vector optimization problem, it seems more reasonable to determine the set of nondominated (Pareto-optimal) solutions instead of finding an optimal solution. For each nondominated vector we may be interested in giving just one or giving the set of all corresponding solutions, leading to the generation of the minimal or the maximal complete set of Pareto-optimal solutions respectively.

Initial definitions of nondominance were given by Pareto (1896). These concepts were used in OR for the first time by Koopmans (1951). An introduction in multi-objective linear programming is given by Zeleny (1974).

Martins (1984a) presents a non-polynomial LS-algorithm for the MinSum multi-objective SP-problem. For the same problem Guerriero & Musmanno (2001) give a non-polynomial LC-algorithm. An extension of Martins' algorithm is presented in Gandibleux *et al.* (2006) where in addition to MinSum, one MaxMin objective is also considered. Multi-objective SP-problems are also examined in Azevedo & Martins (1991), Clímaco & Martins (1981), Corley & Moon (1985), Tung & Chew (1992), Sastry *et al* (2003) and Sastry *et al* (2005).

Hansen (1980) proves that for the MinSum-MinSum bicriterion SP-problem, the cardinality of the minimal complete set increases exponentially with the size of the graph. As a matter of fact, multi-objective SP-problems with more than one MinSum objective are NP-hard (see Gandibleux *et*

al (2006)). This result can be extended for other functions. Clímaco & Martins (1982), Current *et al* (1987) and Tung & Chew (1988) also work with the SP-problem with two MinSum objectives.

Hansen (1980) presents an $O(m^2 \log n)$ algorithm for the bicriterion SP-problem with at least one MinMax/MaxMin bottleneck objective function. The other objective function may either be MinMax/MaxMin or MinSum/MaxSum. Berman *et al* (1990) develop an $O(mp(n))$ algorithm for the MinMax-MinSum bicriterion SP-problem. Consequently, Hansen's algorithm is less efficient for dense graphs. For the same kind of problem Martins (1984b) presents an algorithm with the same complexity as Hansen's.

Here the focus is on the tricriterion SP-problem with at least two bottleneck objective functions. In section 2 an example of an application for the MaxMin-MinMax-MinSum problem is given, followed by the formulation of an $O(m^2 p(n))$ algorithm that is able to produce the minimal complete set of Pareto-optimal solutions. Three different weights $p^1(i,j)$, $p^2(i,j)$ and $p^3(i,j)$ for each arc (i,j), corresponding to each of the three objectives, may be considered. The algorithm orders the weights of the arcs, generating subgraphs with arcs introduced successively so as to satisfy lower/upper bounds for $p^1(i,j)$ and $p^2(i,j)$. For each such subgraph the algorithm searches for a MinSum solution. A new Pareto-optimum is obtained if this solution means an improvement with respect to a set of solutions generated previously. The procedure has some similarities with the algorithm presented by Berman *et al* (1990) but the fact that we consider two bottleneck objectives instead of just one makes it necessary to introduce an additional test. An extension of the algorithm allows the generation of the set of all Pareto-optimal solutions (maximal complete set). Optimality proofs are given.

Section 3 presents computational results for a set of randomly generated problems up to 5000 nodes and section 4 extends some of the previous results closing the paper with the conclusions.

2) The MaxMin-MinMax-MinSum problem:

Let $G = (N,M)$ be a graph with $|N| = n$ and $|M| = m$. Real functions $p^1(i,j)$, $p^2(i,j)$ and $p^3(i,j)$ are associated with each arc $(i,j) \in M$. Let m_1 and m_2 be the quantity of different values assumed by p^1 and p^2 respectively. Without loss of generality let us suppose that the m_1 values $p^1_1, p^1_2, \ldots, p^1_{m_1}$ and the m_2 values $p^2_1, p^2_2, \ldots, p^2_{m_2}$ are arranged in an decreasing/increasing order respectively, i.e., we have $p^1_1 > p^1_2 > \cdots > p^1_{m_1}$ and $p^2_1 < p^2_2 < \cdots < p^2_{m_2}$.

Let $M_{rg} = \{(i,j) \in M \mid p^1(i,j) \geq p^1_r \text{ and } p^2(i,j) \leq p^2_g\}$ and let $G_{rg} = (N, M_{rg})$ be a spanning subgraph of *G*. Of course we have $M_{rg} \supseteq M_{r'g'}$ for $r' \leq r$ and $g' \leq g$. It is also easy to see that $M_{m_1 m_2} = M$.

A path in *G* from $s \in N$ to $t \in N$ is the ordered set $P_{st} = \{s, i_1, i_2, \ldots, i_h, t\}$ where $i_1, i_2, \ldots, i_h \in N$. $AP_{st} = \{a_1, a_2, \ldots, a_{h+1}\}$, with $a_1 = (s, i_1), a_2 = (i_1, i_2), \ldots, a_{h+1} = (i_h, t) \in M$ is the set of arcs belonging to the path. Let S^{st}_G be the set of all paths from *s* to *t* in *G*. In this section we are interested in the following tricriterion shortest path problem:

$$
\begin{aligned}
(P) \quad & \textit{maximize} \quad \min_{(i,j)\in AP_{st}} \{p^1(i,j)\} \\
& \textit{minimize} \quad \max_{(i,j)\in AP_{st}} \{p^2(i,j)\} \\
& \textit{minimize} \quad \sum_{(i,j)\in AP_{st}} p^3(i,j) \\
& \textit{subject to :} \quad P_{st} \in S_G^{st}.
\end{aligned}
$$

For example, in a road transportation problem $p^1(i,j)$ may represent the quality of the road between points *i* and *j*. m_1 labels are available to evaluate this quality and the smaller the label the worse the road. $p^2(i,j)$ measures traffic density and finally $p^3(i,j)$ represents the cost or time of the trip which can be proportional to the distance between *i* and *j*. A transportation company may be interested in simultaneously minimizing costs, traffic jam delays and maximizing the quality of the roads used by their vehicles. We could imagine that there were no ways of reducing all three objectives to a common measure (costs for example). At a first stage the management may be interested in generating all nondominated solutions. At a later stage, those responsible for the route planning may need appropriate criteria for making the final choice either automatically or manually, by inspection. The first option is more attractive in the case of a great number of solutions. It is possible to decrease them by introducing, additionally, upper or lower bounds for some of the objectives. We will be examining this situation in section 4.

The definitions of Pareto-optimum, minimal and maximal complete set follow. Let problem *(P)* be considered for graph *G*. Let $q^{P_{st}} = \left(q_1^{P_{st}}, q_2^{P_{st}}, q_3^{P_{st}}\right) \in \Re^3$ be the *objective vector* associated with path P_{st} where $q_1^{P_{st}} = \min_{(i,j)\in AP_{st}} \{p^1(i,j)\}$, $q_2^{P_{st}} = \max_{(i,j)\in AP_{st}} \{p^2(i,j)\}$ and $q_3^{P_{st}} = \sum_{(i,j)\in AP_{st}} p^3(i,j)$. A path P_{st} *dominates* another path $\overline{P}_{st}$ if $q_1^{P_{st}} \geq q_1^{\overline{P}_{st}}$, $q_2^{P_{st}} \leq q_2^{\overline{P}_{st}}$ and $q_3^{P_{st}} \leq q_3^{\overline{P}_{st}}$ with at least one of the inequalities being satisfied strictly. Following definitions can be made:

(a) P_{st} is a *Pareto-optimum* path if there is no other path $\overline{P}_{st} \in S_G^{st}$ that dominates P_{st}.

(b) A set C^* of different Pareto-optimal solutions is said to be a *minimal complete set* if for any Pareto-optimal solution P_{st} one of the two conditions below apply:

(b1) $P_{st} \in C^*$ and there is no $\overline{P}_{st} \in C^*$, $\overline{P}_{st} \neq P_{st}$ so that $q^{\overline{P}_{st}} = q^{P_{st}}$.

(b2) $P_{st} \notin C^*$ and there is one and only one $\overline{P}_{st} \in C^*$ so that $q^{\overline{P}_{st}} = q^{P_{st}}$.

(c) The set of all Pareto-optimal solutions is called the *maximal complete set*.

It is easy to show that $|C^*| \leq m_1 m_2 \leq m^2 \leq n^4$, i.e., the cardinality of the minimal complete set C^* is polinomially bounded by n^4. From the definition (b) of minimal complete set it should be clear that for each $P_{st} \in C^*$ a different value of the objective vector $q^{P_{st}} = \left(p_r^1, p_g^2, q_3^{P_{st}}\right)$ is associated, meaning that there can be no more elements in C^* than the number of different values of $q^{P_{st}}$. With respect to the first two elements of $q^{P_{st}}$ there can be no more than $m_1 m_2$ different values. But for each pair of values p_r^1, p_g^2, one and only one value $q_3^{P_{st}}$ is possible.

The next step consists in presenting algorithm Min-Max-Sum (*MMS*) which solves problem *(P)* determining a minimal complete set C^*. Of course there may be more than one set of this kind. The algorithm consists of two loops. The outer *i-loop* corresponds to decreasing values of p^1 and the inner *j-loop* corresponds to increasing values of p^2. For each pair of values *i*, *j* a graph G_{ij} is

generated in order to search for an optimal path P_{st} with respect to the MinSum criterion. This is done with the help of *AlgS* at step *3.1.1* and a classical LS or LC-algorithm can be used to obtain the value of $q_3^{P_{st}}$. In order to have a Pareto-optimum, $q_3^{P_{st}}$ has to represent an improvement. This is checked at step *3.1.3* using variable v_k. Q_1, Q_2 and Q_3 are sets containing the values of the three objective functions corresponding to the Pareto-optimal solutions included in C^*.

1. *Given:* $G = (N, M)$, *s, t*
2. *Do:* $C^* := \varnothing$, $Q_1 := \varnothing$, $Q_2 := \varnothing$, $Q_3 := \varnothing$ *and* $v_k := \infty$ *for* $k = 1, 2, \ldots, m_2$
3. *For i from 1 to* m_1 *do*
   - 3.1. *For j from 1 to* m_2 *do*
     - 3.1.1. *Use AlgS to determine the optimal path in* G_{ij}
     - 3.1.2. *Let* P_{st} *be the optimal path. If there is no path from s to t, set* $q_3^{P_{st}} = \infty$
     - 3.1.3. *If* $\left(q_3^{P_{st}} < \min_{1 \le k \le j} \{v_k\} \right)$ *then*
       - 3.1.3.1. $v_j := q_3^{P_{st}}$
       - 3.1.3.2. $C^* := C^* \cup \{P_{st}\}$
       - 3.1.3.3. $Q_1 := Q_1 \cup \{p_i^1\}$, $Q_2 := Q_2 \cup \{p_j^2\}$ *and* $Q_3 := Q_3 \cup \{q_3^{P_{st}}\}$
   - 3.2. *End*
4. *End*

It is easy to show that the complexity of algorithm *MMS* is $O(m^2 p(n))$. Two loops of size m_1 and m_2, with $m_1, m_2 \le m$ lead to m^2 iterations in the worst case. Each iteration uses *AlgS* of complexity $p(n)$. For example, if the graph has non-negative weights an n^2-Dijkstra algorithm may be used for *AlgS*, i.e. $p(n) = n^2$, and the result is an $O(m^2 n^2)$ algorithm for *MMS*. We are supposing that $p(n) \ge n^2$ (step *3.1.3* is $O(n^2)$).

Next it will be proved that C^* is a minimal complete set. Theorem 1 proves that all P_{st} generated by algorithm *MMS* are Pareto-optimal solutions. Theorem 2 uses theorem 1 to prove that C^* is a minimal complete set. Let iteration $i = r$ and $j = g$ be called the *rg-iteration*.

Theorem 1: *Let* P_{st} *be the path inserted in* C^* *at the rg-iteration of algorithm MMS. Following two facts are true:*

(a) $q_1^{P_{st}} = p_r^1$ *and* $q_2^{P_{st}} = p_g^2$;

(b) P_{st} *is a Pareto-optimal solution of (P).*

Proof:

(a) It will be shown by contradiction that the values p_r^1 and p_g^2 associated to P_{st} by algorithm *MMS* are effectively the values $q_1^{P_{st}}$ and $q_2^{P_{st}}$ of the two bottleneck functions MaxMin and MinMax. Since P_{st} was obtained for G_{rg} we have $p^1(l,h) \ge p_r^1$ and $p^2(l,h) \le p_g^2$ for all $(l,h) \in AP_{st}$. It follows that $q_1^{P_{st}} \ge p_r^1$ and $q_2^{P_{st}} \le p_g^2$. Suppose that $q_1^{P_{st}} = p_{r'}^1 > p_r^1$ and

$q_2^{P_{st}} = p_{g'}^2 \le p_g^2$ (the other case can be proven in a similar way). Consequently, all arcs $(l,h) \in AP_{st}$ have $p^1(l,h) \ge p_{r'}^1$ and $p^2(l,h) \le p_{g'}^2$. By definition, it follows that $r' < r$ and $g' \le g$. Hence P_{st} is a path in graph $G_{r'g'}$ at the *r'g'-iteration* preceding the *rg-iteration*. This leads to a contradiction because step *3.1.3* guarantees that if P_{st} is a path at the *r'g'-iteration* it could never have been included in C^* at the *rg-iteration*. Thus, it can be stated that $q_1^{P_{st}} = p_r^1$. A similar reasoning proves $q_2^{P_{st}} = p_g^2$.

(b) Again the proof is by contradiction. Suppose that P_{st} is not a Pareto-optimal solution. Then, by definition, there exists a path $\hat{P}_{st}$ that dominates P_{st}. Using *(a)* it can be said that $q_1^{\hat{P}_{st}} \ge q_1^{P_{st}} = p_r^1$, $q_2^{\hat{P}_{st}} \le q_2^{P_{st}} = p_g^2$ and $q_3^{\hat{P}_{st}} \le q_3^{P_{st}}$ with at least one of the inequalities being satisfied strictly. Without loss of generality consider the case where $q_1^{\hat{P}_{st}} = p_{r'}^1 > q_1^{P_{st}} = p_r^1$ and $q_2^{\hat{P}_{st}} = p_{g'}^2 \le q_2^{P_{st}} = p_g^2$. The proof of the other cases is similar. From the initial assumptions of this section it follows that $r' < r$, $g' \le g$ and $\hat{P}_{st}$ is a path in graph $G_{r'g'}$ at the *r'g'-iteration* preceding the *rg-iteration*. Due to step *3.1.3* at the end of the *r'g'-iteration*, there must exist a path $\overline{P}_{st} \in C^*$ better or equal than $\hat{P}_{st}$. $\overline{P}_{st}$ may have been generated at the *r'g'-iteration* or may have been generated at a previous iteration. Of course it is possible that $\overline{P}_{st} = \hat{P}_{st}$. Using the inequality from the beginning of the paragraph it can be stated that $q_3^{\overline{P}_{st}} \le q_3^{\hat{P}_{st}} \le q_3^{P_{st}}$. On the other side, step *3.1.3* used at the *rg-iteration* guarantees that $q_3^{P_{st}} < \min_{1 \le k \le g} \{v_k\} \le \min_{1 \le k \le g'} \{v_k\} \le q_3^{\overline{P}_{st}}$ leading to a contradiction.

■

Theorem 2: *Algorithm MMS produces, upon completion, a minimal complete set C^* of Pareto-optimal solutions of problem (P).*

Proof: Part *(b)* of theorem 1 shows that C^* is a set of Pareto-optimal solutions. Part *(a)* shows that any two solutions P_{st} and $\hat{P}_{st}$ included in C^* by algorithm *MMS* are different with $q^{P_{st}} \ne q^{\hat{P}_{st}}$. Thus, part *(b1)* of the definition of minimal complete set is satisfied.

Next, part *(b2)* of the definition will be examined. Let $P_{st} \notin C^*$ be a Pareto-optimal solution so that $q_1^{P_{st}} = p_r^1$ and $q_2^{P_{st}} = p_g^2$. P_{st} is a path in graph G_{rg}. Therefore there exists a path $\hat{P}_{st}$ obtained at step *3.1.2* of the *rg-iteration*. The fact that $\hat{P}_{st}$ was obtained in graph G_{rg}, guarantees that $p^1(l,h) \ge p_r^1$ and $p^2(l,h) \le p_g^2$ for all $(l,h) \in A\hat{P}_{st}$. This results in $q_1^{\hat{P}_{st}} \ge p_r^1 = q_1^{P_{st}}$ and $q_2^{\hat{P}_{st}} \le p_g^2 = q_2^{P_{st}}$. Furthermore, optimality of *AlgS* guarantees that $q_3^{\hat{P}_{st}} \le q_3^{P_{st}}$. The fact that P_{st} is a Pareto-optimal solution assures that none of these inequalities can be satisfied strictly, leading to $q^{P_{st}} = q^{\hat{P}_{st}}$. Thus, $\hat{P}_{st}$ is also a Pareto-optimal solution.

It is easy to show that at step *3.1.3* of the *rg-iteration* the condition $q_3^{\hat{P}_{st}} < \min_{1 \le k \le g} \{v_k\}$ is satisfied, leading to $\hat{P}_{st} \in C^*$. If $q_3^{\hat{P}_{st}} \ge \min_{1 \le k \le g} \{v_k\}$ there would exist a path $\overline{P}_{st} \in C^*$, inserted at a

previous *r'g'-iteration,* so that $q_3^{\overline{P}_{st}} \leq q_3^{\hat{P}_{st}}$. Evidently $r' \leq r$ and $g' \leq g$, with at least one of the inequalities being satisfied strictly. Thus, $\hat{P}_{st}$ would be dominated by $\overline{P}_{st}$, since *(a)* of theorem 1 guarantees that $q_1^{\overline{P}_{st}} = p_{r'}^1 \geq p_r^1 = q_1^{\hat{P}_{st}}$ and $q_2^{\overline{P}_{st}} = p_{g'}^2 \leq p_g^2 = q_2^{\hat{P}_{st}}$, with at least one of the inequalities being satisfied strictly. Consequently $\hat{P}_{st}$ would not be a Pareto-optimal solution leading to a contradiction. Hence $\hat{P}_{st}$ satisfies test *3.1.3* and is inserted in C^* satisfying *(b2)*.

■

To generate the maximal complete set it is just necessary to replace *AlgS* by an algorithm which generates the set of all optimal solutions at step *3.1.1* and then make the necessary adjustments. One should remember that the cardinality of this set increases exponentially with the size of the problem (in the worst case all paths are optimal). Thus, an algorithm to generate the maximal complete set is necessarily non-polynomial (NP) (see also Ehrgott and Gandibleux (2000)).

3) Computational results:

This section presents computational results for a set of randomly generated problems. Each line of table 1 represents the average values of three problems. Nine classes of problems $Prob_{ij}$ for *i, j = 1, 2* and *3* are generated. *i = 1, 2* and *3* represents a graph with *50*, *1000* and *5000* nodes respectively. *j = 1, 2* and *3* represents increasing graph densities. For each pair of nodes $l, h \in N$ an integer *k* is generated randomly between *1* and α. If *k=1* then $(l,h) \in M$. Otherwise, there will be no corresponding arc associated to the graph. *j = 1, 2* and *3* corresponds to $\alpha = 10, 2$ and *1* respectively. Of course for $\alpha = 1$ the graph is complete. For each class, three kinds of problems are produced with respect to values for p^1 and p^2, generating integers randomly for the interval *[1, 5]*, *[1, 20]* and *[1, 70]* respectively. With respect to p^3, integers are generated randomly between *1* and *10000* for each of the three cases.

Problems were run on a PC with a *CENTRINO CORE DUO* processor at *1.6 GHz* with *1024 Mb* of *RAM*. Programming language was *C* and a heap implementation of Dijkstra's algorithm was used for *AlgS*.

At table 1 *cms* and *cMs* are the average cardinality of the minimal and maximal complete set respectively. $time_{cms}$ and $time_{cMs}$ are the average computer times in seconds that were necessary to generate the corresponding sets.

A first comment relates to the values obtained for *cms* and *cMs* as well as the corresponding $time_{cms}$ and $time_{cMs}$. Very similar values were obtained. This may be due to the special data which was used, particularly the values of p^3 which were generated from a rather large interval, producing few paths with the same length. An experience was made, generating p^3 from a *[1, 10]* interval and the results were quite different. Again, nine problems were considered for each class. Instead of having *cMs/cms = 1.0* as in table 1 the results for the new set of problems were $1.0 \leq cMs/cms \leq 5.9$. For example, the average values for the class $Prob_{23}$ with $m_1 = m_2 = 5$ were *cms = 6* and *cMs = 16*.

Another more important conclusion refers to the small increase of *cms* or *cMs* with respect to increasing values of *m* and *n*. Comparing the problems obtained for *n = 1000* with the problems obtained for *n = 5000* we see that almost the same number of Pareto-optimal solutions was obtained. Even if we compare *n = 50* with *n = 1000*, representing an increase of the size of the graph of approximately 20 times, the corresponding increase of the average *cms* of nearly 6 times is far below. These conclusions are reinforced comparing increasing values of *m* for a certain fixed

Class	*n*	*m*	$m_1 = m_2$	*cms*	$time_{cms}$	*cMs*	$time_{cMs}$
$Prob_{11}$	50	259	5	6	0	6	0
$Prob_{11}$	50	237	20	8	0	8	0
$Prob_{11}$	50	254	69	4	0	4	0
$Prob_{12}$	50	1230	5	12	0	12	0
$Prob_{12}$	50	1219	20	32	0	32	0
$Prob_{12}$	50	1222	70	40	0	40	0
$Prob_{13}$	50	2450	5	15	0	15	0
$Prob_{13}$	50	2450	20	34	0	34	0
$Prob_{13}$	50	2450	70	60	0	60	0
$Prob_{21}$	1000	99942	5	21	0	21	0
$Prob_{21}$	1000	99885	20	103	2	103	2
$Prob_{21}$	1000	99826	70	281	32	281	32
$Prob_{22}$	1000	499567	5	17	0	17	0
$Prob_{22}$	1000	499544	20	81	3	81	3
$Prob_{22}$	1000	499558	70	304	53	304	54
$Prob_{23}$	1000	999000	5	19	0	19	0
$Prob_{23}$	1000	999000	20	113	5	114	5
$Prob_{23}$	1000	999000	70	322	191	323	208
$Prob_{31}$	5000	2499672	5	21	5	21	5
$Prob_{31}$	5000	2500231	20	135	62	135	62
$Prob_{31}$	5000	2500726	70	394	892	394	901
$Prob_{32}$	5000	12497615	5	20	6	21	7
$Prob_{32}$	5000	12497449	20	152	115	154	119
$Prob_{32}$	5000	12497604	70	433	1246	435	1282
$Prob_{33}$	5000	24995000	5	20	7	21	7
$Prob_{33}$	5000	24995000	20	124	97	126	103
$Prob_{33}$	5000	24995000	70	455	1529	458	1616

Table 1: *Average values of times and the number of Pareto-optimal solutions.*

value of *n*. With exception of the very small problems ($n = 50$ and $m \approx 250$) where the sparsity of the graph led to a very small number of Pareto-optimal solutions, for a certain fixed value of *n,* there is almost no change of *cms* with increasing values of *m*.

A greater sensitivity is verified when comparing the variation of *cms* or *cMs* with respect to m_1 or m_2. With exception of the very small problems, one is tempted to establish an almost linear relationship between *cms* or *cMs* and m_1 or m_2. Anyway, the values obtained for *cms* or *cMs* seem quite far from the upper limit $m_1 m_2$. This is specially true for bigger values of m_1 or m_2.

A general conclusion concerning the variation of *cms* or *cMs* is that the number of Pareto-optimal solutions seems to increase rather slowly with *n*, *m,* m_1 or m_2. At this stage it is quite hard to tell if this result can be generalized or if it relates strongly to the kind of data which was used.

4) Extensions and conclusions:

The tricriterion SP-problem with at least two bottleneck objective functions, gives rise to a series of different problems. If order does not matter, three problems arise with respect to the two bottleneck functions MaxMin and MinMax: MaxMin-MinMax, MaxMin-MaxMin, MinMax-MinMax. Now consider the six possibilities for the third objective function: MaxMin, MinMax, MaxSum, MinSum, MaxProd and MinProd. The result is a set of eighteen similar problems which can be solved by the algorithm *MMS* with some minor adjustments.

A different problem arises if we introduce additional restrictions into problem *(P)*. An easy case happens when these restrictions are of the bottleneck type, i.e., we require the maximum or minimum weight of a path to be respectively *lesser than* or *greater than* a certain value. In this case it is sufficient to delete the corresponding arcs from the graph (see Berman *et al.* (1990)). Considering an upper bound for the sum of the weights p^3 of the path is also easy to solve if we work with the MinSum objective. If the solution obtained by *AlgS* for a certain value of *i* and *j* does not satisfy the upper bound it will not be included in the minimum complete set. A similar comment can be made if we consider a lower bound for a MaxSum objective. A more difficult problem results from taking another weight, different from p^3, to evaluate these bounds.

A particular case for algorithm *MMS* occurs when $p^1 = p^2$ where $p^1 = p^2 = p^3$ is just a special situation. Problems represented by a graph with just one weight per arc, where paths need to be evaluated from several different perspectives, may fall under this case. The efficiency of algorithm *MMS* is improved and approximately half of the number of iterations is needed to generate the minimal complete set.

An extension for the tricriterion spanning tree problem is also possible. For the MaxMin-MinMax-MinSum case it is sufficient to change at step *3.1.1* of algorithm *MMS* the *AlgS* by a traditional minimum spanning tree algorithm like *Prim* or *Kruskal.* If instead of MinSum we have another MinMax objective function it is sufficient to consider algorithms like those presented in Camerini (1978) or Gabow & Tarjan (1988).

A last word should be said with respect to practical aspects of problem *(P)* and algorithm *MMS*. As already seen, algorithm *MMS* needs $m_1 m_2$ iterations to generate C^*. Hence, for big values of m_1 and m_2, there may be a lot of computational work involved. For a *PC*, as in our case, it seems reasonable to have $m_1, m_2 \leq 100$. As a matter of fact, for great values of m_1 and m_2, the determination of C^* seems not to make much sense. A great number of Pareto-optima may leave the decision maker in a difficult situation having to choose among a rather large set of possible solutions. In contrast, for small values of m_1 and m_2 the number of Pareto-optima seems to increase very slowly with the size of the graph meaning that big graphs can be treated quite effectively.

References:

Ahuja RK, Magnanti TL, Orlin JB. Network flows. New Jersey; 1993.

Azevedo JA, Martins EQV. An algorithm for the multiobjective shortest path problem on acyclic networks. Investigação Operacional 1991; 11 (1); 52-69.

Batta R, Chiu SS. Optimal obnoxious paths on a network: Transportation of hazardous materials. Operations Research 1988; 36; 84-92.

Bellman RE. On a routing problem. Quarterly of Applied Mathematics 1958; 16; 87-90.

Berman O, Einav D, Handler G. The constrained bottleneck problem in networks. Operations Research 1990; 38; 178-181.

Camerini PM. The min-max spanning tree problem and some extensions. Information Processing Letters 1978; 7; 10-14.

Clímaco JCN, Martins EQV. On the determination of the nondominated paths in a multiobjective network problem. Methods in Operations Research 1981; 40; 255-258.

Clímaco JCN, Martins EQV. A bicriterion shortest path algorithm. European Journal of Operational Research 1982; 11; 399-404.

Corley HW, Moon ID. Shortest paths in networks with vector weights. Journal of Optimization Theory and Applications 1985; 46; 79-86.

Current JR, Min H. Multiobjective design of transportation networks: Taxonomy and annotation. European Journal of Operational Research 1986; 26; 187-201.

Current JR, Revelle CS, Cohon JL. The median shortest path problem: A multiobjective approach to analyze cost vs. accessibility in the design of transportation networks. Transportation Science 1987; 21 (3); 188-197.

Current JR, Marsh M. Multiobjective transportation network design and routing problems: Taxonomy and annotation. European Journal of Operational Research 1993; 103; 426-438.

Dijkstra EW. A note on two problems in connexion with graphs. Numer. Math. 1959; 1; 269-271.

Ehrgott M, Gandibleux X. A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spektrum 2000; 22; 425-460.

Gabow HN, Tarjan RE. Algorithms for two bottleneck optimization problems. Journal of Algorithms 1988; 9; 411-417.

Gandibleux X, Beugnies F, Randriamasy S. Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function. A Quarterly Journal of Operations Research 2006; 4; 47-59.

Guerriero F, Musmanno R. Label correcting methods to solve multicriteria shortest path problems. Journal of Optimization Theory and Applications 2001; 111 (3); 589-613.

Gondran M, Minoux M. Graphs and Algorithms. John Wiley; 1986.

Hansen P. Bicriterion path problems. In: Fandel G, Gal T (Eds), Multicriteria decision making: Theory and applications. Lecture Notes in Economics and Mathematical Systems, vol. 177. Springer, Heidelberg; 1980. p. 109-127.

Koopmans TC. Activity analysis of production and allocation. John Wiley & Sons, New York; 1951.

Martins EQV. On a multicriteria shortest path problem. European Journal of Operational Research 1984a; 16; 236-245.

Martins EQV. On a special class of bicriterion path problems. European Journal of Operational Research 1984b; 17; 85-94.

Pareto V. Course d'economic politique. Lausanne, Rouge; 1896.

Pinto LL, Bornstein CT. Software CamOtim. Available at: http://www.cos.ufrj.br/camotim; 2006a.

Pinto LL, Bornstein CT. Algoritmos para problemas de caminho ótimo em grafos (Algorithms for optimal path problems in graphs). Proceedings of the XXXVIII Brazilian Symposium on Operations Research. Goiânia-GO; 2006b; 2418-2419.

Sastry VN, Janakiraman TN, Mohideen SI. New algorithms for multi objective shortest path problem. Opsearch 2003; 40; 278-298.

Sastry VN, Janakiraman TN, Mohideen SI. New polynomial time algorithms to compute a set of Pareto optimal paths for multi-objective shortest path problems. International Journal of Computer Mathematics 2005; 82; 289-300.

Tung CT, Chew KL. A bicriterion Pareto-optimal path algorithm. Asia-Pacific Journal of Operational Research 1988; 5; 166-172.

Tung CT, Chew KL. A multicriteria Pareto-optimal path algorithm. European Journal of Operational Research 1992; 62; 203-209.

Zeleny M. Linear Multiobjective Programming. Lecture Notes in Economics and Mathematical Systems, vol. 95. Springer-Verlag; 1974.

Anexo 2

Este anexo contém o artigo de Pinto e Pascoal intitulado *On algorithms for the tricriteria shortest path problem with two bottleneck objective functions* e submetido à *Computers & Operations Research*. A versão inicial foi submetida em março de 2009. Em maio recebemos as sugestões dos revisores e as implementamos na atual versão. Este anexo contém essa nova versão, submetida em setembro de 2009.

On algorithms for the tricriteria shortest path problem with two bottleneck objective functions

Leizer Lima Pinto[1,2] and Marta M. B. Pascoal*[3,4]

[1]COPPE/UFRJ - Federal University of Rio de Janeiro, Brazil
Department of Systems Engineering and Computer Science
E-mail: leizer@cos.ufrj.br

[2]Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[3]Departamento de Matemática da Universidade de Coimbra
Apartado 3008, EC Universidade, 3001-454 Coimbra, Portugal
E-mail: marta@mat.uc.pt

[4]Institute for Systems and Computers Engineering – Coimbra (INESCC)

Submitted March 2009
Revised September 2009

Abstract

This paper addresses a tricriteria path problem involving two bottleneck objective functions and a cost. It presents an enhanced method that computes shortest paths in subnetworks, obtained by restricting the set of arcs according to the bottleneck values in order to find the minimal complete set of Pareto-optimal solutions, and taking into account the objective values of the determined shortest paths to reduce the number of considered subnetworks, and thus the number of solved shortest path problems. A labeling procedure for the problem is also developed. The algorithms are compared with the previous literature. Moreover a variant of the first method is presented. Its aim is to choose the solutions with the best bottleneck value when the cost is the same. Results for random instances reveal that the enhanced method is the fastest, and that, in average, it runs in less than 20 seconds for networks with 30 000 nodes, an average degree of 20 and 1 000 distinct bottleneck values. Its variant that avoids ties improved the former version in up to 15% for costs in $[1, 10]$.

Keywords: Tricriteria path problem, cost function, bottleneck function, Pareto-optimal solution.

1 Introduction

The shortest path problem, as well as variants involving other objective functions rather than the distance or the cost of the paths, is a classical problem that has been studied intensively since the 1950's. The question of finding the best route(s) between two points with several objective functions has also been a topic of research for many authors. In 1980 Hansen [9] presented theoretical results and labeling algorithms for a list of bicriteria path problems, including the bicriteria shortest path problem and the shortest path problem with a bottleneck objective function. Since then other forms of labeling algorithms have been designed for the bicriteria shortest path problem, for instance in the works by Brumbaugh-Smith and Shier [3] or by Skriver and Andersen [19]. At the same time

*Corresponding author.

a method that ranks shortest paths in order to determine non-dominated bicriteria shortest paths was introduced by Clímaco and Martins [5], while Mote *et al.* derived a two phase method for the same problem [13]. A recent comparison between some of these methods can be found in [18]. Fewer works deal with the shortest path problem when there are more than two criteria, but some of the aforementioned approaches have been extended for this case. Labeling algorithms for the multicriteria shortest path problem have been proposed by Martins [10], Guerriero and Musmanno [8], and Tung and Chew [20], while a ranking method was derived by Clímaco and Martins [4].

As for the shortest path problem with an additional bottleneck objective function, Martins [11] developed algorithms for finding maximal and minimum sets of solutions. More recently this algorithm was generalized by Gandibleux *et al.* [7] in order to deal with more than a single cost function and exactly one of bottleneck type.

The present paper deals with path problems involving two bottleneck functions, either of MaxMin or of MinMax type, and one additive cost function. Polynomial algorithms for path problems considering cost and bottleneck functions have been presented by Hansen [9], Martins [11] and Berman *et al.* [2]. More recently the tricriteria path problem with a cost function and two bottleneck functions was analyzed by Pinto *et al.* [14, 15]. In these cases the goal is the generation of a set of paths, all having Pareto-optimal objective values. The finite number of values that a bottleneck function can have yields algorithms with polynomial complexity order. The focus of this paper is to describe an algorithm for improving the method introduced in [14] and modified later in [15]. Afterwards a labeling approach is designed. We start by introducing some notation and by formulating the problem itself.

Consider a graph $G = (\mathcal{N}, \mathcal{A})$ with $|\mathcal{N}| = n$ and $|\mathcal{A}| = m$. For each arc $(i, j) \in \mathcal{A}$, let $c_{ij}^k \in \mathbb{R}$ be its weights, $k = 1, 2, 3$. Given initial and terminal nodes in $\mathcal{N}$, s and t, let $p = \langle i_1 = s, i_2, \ldots, i_\ell = t \rangle$, with $(i_k, i_{k+1}) \in \mathcal{A}$ for $k = 1, \ldots, \ell - 1$, denote a path in G. For simplicity we write $i \in p$ if i is a node in the sequence p, and $(i, j) \in p$ if i immediately precedes j in p. Let $\mathcal{P}$ stand for the set of paths from s to t in G.

As mentioned above we deal with two bottleneck functions and one cost function, therefore the *objective vector* associated with path p is given by $c(p) = (c_1(p), c_2(p), c_3(p)) \in \mathbb{R}^3$, where

$$c_1(p) = \max_{(i,j)\in p} \{c_{ij}^1\}, \quad c_2(p) = \max_{(i,j)\in p} \{c_{ij}^2\} \quad \text{and} \quad c_3(p) = \sum_{(i,j)\in p} c_{ij}^3.$$

For the sake of simplicity the bottleneck functions are considered as of MinMax type, yet there is no loss of generality in doing so. The tricriteria shortest path problem with two bottleneck functions (TSPPB) is then defined as

$$\min\{c(p) : \ p \in \mathcal{P}\}.$$

Possible applications of the TSPPB in transportation and in communication networks can be found in [14, 16]. The first case focuses a road network the arcs of which are associated with a measure of the road quality, c_{ij}^1, the traffic density, c_{ij}^2, and the cost or time for traveling from i to j, c_{ij}^3. The aim is to find a path between s and t, which maximizes the quality while it minimizes the traffic jams and the costs. The second addresses the transmission of information between two nodes in a sensors network. In this case c_{ij}^1 is the power available at the battery represented by node i, c_{ij}^2 is the average waiting time at that node, that is the time the information is held in i or

a congestion measure related with i, and c^3_{ij} the transmission cost along (i,j). Based on this model a telecommunications company may be interested in finding a route from s to t, which maximizes the minimum power available in the batteries and minimizes the waiting time as well as the total cost.

It is said that $p \in \mathcal{P}$ *dominates* $\bar{p} \in \mathcal{P}$ when $c(p) \leq c(\bar{p})$ and $c(p) \neq c(\bar{p})$. A path $p \in \mathcal{P}$ is *Pareto-optimal* if it is not dominated by any other path in $\mathcal{P}$. Similarly $c(p)$ dominates $c(\bar{p})$ when p dominates $\bar{p}$, for $p, \bar{p} \in \mathcal{P}$, and $c(p)$, $p \in \mathcal{P}$, is said to be Pareto-optimal if there is no other path $\bar{p} \in \mathcal{P}$ such that $c(\bar{p})$ dominates $c(p)$. A set $\mathcal{P}^* \subseteq \mathcal{P}$ of Pareto-optimal solutions is a *minimal complete set* if for each $p, q \in \mathcal{P}^*$ we have $c(p) \neq c(q)$ and for any Pareto-optimal solution $p^* \in \mathcal{P}$ there exists $\bar{p} \in \mathcal{P}^*$ so that $c(p^*) = c(\bar{p})$.

This paper contains three other sections. The next one introduces an algorithm to compute a minimal complete set as well as the use of a shortest path method that avoids ties in the costs in order to reduce the number of subroutine calls. It also presents a labeling algorithm for the same problem. Section 3 presents an algorithmic analysis. Finally, Section 4 reports and discusses computational results.

2 Algorithms for the TSPPB

The methods initially proposed to find the minimal complete set for the TSPPB, first `MMS` and later `MMS-R`, [14, 15], use the fact that fixing bounds on the two bottleneck functions, c_1, c_2, produces a subgraph of G. Therefore, Pareto-optimal solutions for this problem are amongst the shortest paths in each of these subgraphs. `MMS` [14] determines the shortest path in subgraphs obtained by fixing all possible combinations of bounds on c_1 and c_2. In [15] the same procedure is followed but the bounds are fixed by decreasing order, which allows algorithm `MMS-R` to skip infeasible subproblems. Namely, if no path is found on a certain subgraph, then the first cost bound can be set to the next value to observe. In this section a new algorithm is proposed for the TSPPB. It is still based on the computation of shortest paths in subgraphs of G, but the number of subproblems to be solved is reduced by taking into account the objective values of the solutions that are obtained. A modification of this method aimed at decreasing the number of shortest path problems is also proposed, as well as a label setting method.

Let m_k be the number of different values of c^k_{ij}, $(i,j) \in \mathcal{A}$, and suppose these are arranged in decreasing order, i.e., $c^k_1 > c^k_2 > \cdots > c^k_{m_k}$, $k = 1, 2$. Given $v = (v_1, v_2)$, with $v_k \in \{1, \ldots, m_k\}$ and $k = 1, 2$, considering the subset of arcs

$$\mathcal{A}_v = \{(i,j) \in \mathcal{A} : \ c^k_{ij} \leq c^k_{v_k}, \ k = 1, 2\},$$

the subgraph of G, $G_v = (\mathcal{N}, \mathcal{A}_v)$, can be defined.

Because our purpose is to find a minimal complete set, that is, one solution for each Pareto-optimal triple of objective values, there is at most one solution to be considered for each pair (v_1, v_2) of (c_1, c_2) values. If such a solution exists it is the shortest path in the subgraph defined by (v_1, v_2). The presentation is simplified if the set of subgraphs G_v is represented as an $m_1 \times m_2$ matrix, denoted by C and used to store the several obtained shortest paths.

The method below works in two phases. One solves shortest path problems and stores the solutions at a certain position of C. This phase is followed by another, for filtering possible dominated or equivalent solutions in C.

Blocks method The aim of this new version is to use the objective values of the computed shortest paths, in order to skip some subproblems, i.e., some positions in matrix C. A similar idea was exploited in [11] when dealing with only two criteria, one bottleneck and one additive. It is now extended for one more objective function.

In the first phase, for each value c_h^1 fixed as a bound of c_{ij}^1, decreasing bounds are chosen for c_{ij}^2. The vector $v = (v_1, v_2)$ such that $c_k(p) = c_{v_k}^k$, $k = 1, 2$, is called the *final position of* p. A binary matrix S, with dimension $m_1 \times m_2$, marks the subproblems (or iterations) that have to be solved, that is, the shortest path problem in G_v is solved if and only if $S_v = 1$; otherwise the next position in C is considered. The solutions that are obtained are inserted in the final position in matrix C. Moreover, given p the shortest path in $G_{\bar{v}}$ and v its final position, then $S_{\hat{v}}$ is set to 0 for every $\hat{v}$ such that $\bar{v} \leq \hat{v} \leq v$.

The matrix C resulting from the first phase of the algorithm may contain some dominated or equivalent paths, solutions that are eliminated during the second phase. Let Q be an $m_1 \times m_2$ matrix that stores the c_3 values of the solutions obtained at the corresponding positions. When phase 1 is over, given the positions v and $\bar{v}$ in C, such that $v \leq \bar{v}$ and $v \neq \bar{v}$, the correspondent paths p and $\bar{p}$ satisfy $c_3(p) < c_3(\bar{p})$, therefore p is not dominated by $\bar{p}$. The solutions are thus filtered by checking all the objective values of the paths stored in C, that is, the values in matrix Q.

Algorithm 1 outlines the blocks method in pseudo-code language.

Algorithm 1. *Finding a minimal complete set of Pareto-optimal paths with blocks*

```
For i = 1, ..., m1 Do
    For j = 1, ..., m2 Do C_ij ← ∅; Q_ij ← +∞; S_ij ← 1
ctrl ← m2; i ← 1
Phase 1. Computing Pareto-optimal candidates
While i ≤ m1 and ctrl ≠ 0 Do
    j ← 1
    While j ≤ ctrl Do
        If S_ij = 1 Then
            p ← shortest path in G_ij in terms of c3
            If p is not defined Then ctrl ← j − 1
            Else
                v_k ← indexes such that c_k(p) = c^k_{v_k}, k = 1, 2
                If c3(p) < Q_{v1 v2} Then
                    C_{v1 v2} ← p; Q_{v1 v2} ← c3(p)
                    For g = i, ..., v1 Do
                        For h = j, ..., v2 Do S_gh ← 0
        j ← j + 1
```

$i \leftarrow i + 1$

Phase 2. Filtering Pareto-optimal solutions in C

`For` $i = m_1, \ldots, 1$ `Do`

`For` $j = m_2, \ldots, 1$ `Do`

`If` $Q_{ij} \neq +\infty$ `Then`

`If` $j > 1$ `and` $Q_{ij} \leq Q_{ij-1}$ `Then` $C_{ij-1} \leftarrow \emptyset$; $Q_{ij-1} \leftarrow Q_{ij}$

`If` $i > 1$ `and` $Q_{ij} \leq Q_{i-1j}$ `Then` $C_{i-1j} \leftarrow \emptyset$; $Q_{i-1j} \leftarrow Q_{ij}$

Alternative method Aimed at reducing the number of shortest path problems that have to be solved, the routine used by the blocks method can be replaced by a variant, adapted from the algorithm described in [12], that chooses the best option for labeling whenever there is a tie in the cost. This procedure is more demanding than a regular shortest path algorithm, as it implies storing and updating the bottleneck function values, besides the additive cost. Still it can avoid the computation of certain paths and, for some cases, outperforms the previous version, as the empirical tests reported in Section 4 show. Denoting by π_i^k the value of c_k associated with a path from s to a node i at a certain step of this labeling algorithm, $k = 1, 2, 3$, the shortest path algorithm variant consists in rewriting the labeling test as:

`If` $(\pi_j^3 > \pi_i^3 + c_{ij}^3)$ `or` $(\pi_j^3 = \pi_i^3 + c_{ij}^3$ `and` $(\pi_j^1 > \max\{\pi_i^1, c_{ij}^1\}$ `or`
$(\pi_j^1 = \max\{\pi_i^1, c_{ij}^1\}$ `and` $\pi_j^2 > \max\{\pi_i^2, c_{ij}^2\})))$ `Then`

Node j is labeled using arc (i, j)

$\pi_j^1 \leftarrow \max\{\pi_i^1, c_{ij}^1\}$; $\pi_j^2 \leftarrow \max\{\pi_i^2, c_{ij}^2\}$; $\pi_j^3 \leftarrow \pi_i^3 + c_{ij}^3$

Labeling method A label setting approach for the `TSPPB`, a tricriteria extension of the Dijkstra's algorithm, can also be designed to generate a minimal complete set of Pareto-optimal paths. This approach uses two different sets of labels for each network node, the set of permanent labels and the set of temporary labels. A label l_p^j is associated with a path p from the source node s to node j, this label can be represented as $l_p^j = [c_1(p), c_2(p), c_3(p), i, h]$, where i is the node immediately preceding j in p and h is the position, in the list of labels of i, of the label that produced l_p^j. At the end of the algorithm, the values h can be used to retrieve the solutions.

At each iteration the lexicographically smallest amongst all temporary labels is selected and converted into a permanent label. Let i be the node associated with this label, and p be the correspondent path from s to i. For each $(i, j) \in \mathcal{A}$ a new label l_q^j is created, such that $l_q^j = \left[\max\{c_1(p), c_{ij}^1\}, \max\{c_2(p), c_{ij}^2\}, c_3(p) + c_{ij}^3, i, h\right]$. This label is submitted to the dominance test considering the two sets of labels of node j. If l_q^j is not dominated by any of those labels, then it is inserted in the temporary set of j and the labels in this set that are dominated by l_q^j are deleted.

3 Algorithms analyses

In this section the blocks method is illustrated. Also the correction of this algorithm is proved and the complexities of this approach and of the label setting approach are analyzed. Let us consider the graph G depicted in Figure 1. For this instance of the TSPPB $m_1 = 5$ and $m_2 = 4$, with $c_{ij}^1 \in \{9, 7, 6, 3, 2\}$ and $c_{ij}^2 \in \{8, 5, 4, 3\}$, for any $(i, j) \in \mathcal{A}$.

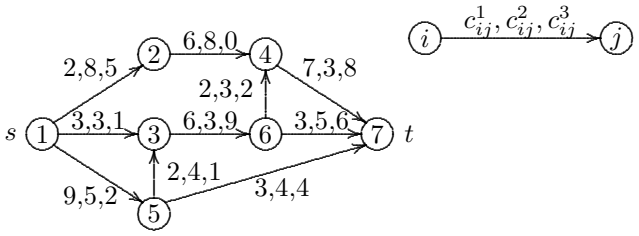


Figure 1: Graph G

Table 1 shows a list of the shortest paths and the matrix C obtained when applying to G Algorithm 1. Columns p, v and IP present the computed paths, the iteration for obtaining them and their insertion positions, respectively. The symbols '$\emptyset$' and '$-$' mean, respectively, there is no path in graph G_v, therefore no new solution is found at iteration v, and the shortest path problem was not solved in that iteration, that is $S_v = 0$.

p	c_1	c_2	c_3	v	IP	S_v
$p_1 = \langle 1,5,7\rangle$	9	5	6	(1,1)	(1,2)	1
$-$	$-$	$-$	$-$	(1,2)	$-$	0
$p_2 = \langle 1,3,6,4,7\rangle$	7	3	20	(1,3)	(2,4)	1
$-$	$-$	$-$	$-$	(1,4)	$-$	0
$p_3 = \langle 1,2,4,7\rangle$	7	8	13	(2,1)	(2,1)	1
$p_4 = \langle 1,3,6,7\rangle$	6	5	16	(2,2)	(3,2)	1
$-$	$-$	$-$	$-$	(2,3)	$-$	0
$-$	$-$	$-$	$-$	(2,4)	$-$	0
$p_5 = \langle 1,3,6,7\rangle$	6	5	16	(3,1)	$-$	1
$-$	$-$	$-$	$-$	(3,2)	$-$	0
$\emptyset$	$-$	$-$	$-$	(3,3)	$-$	1
$\emptyset$	$-$	$-$	$-$	(4,1)	$-$	1

C:

	1	2	3	4
1		p_1		
2	p_3			p_2
3		p_4		
4				
5				

Table 1: Result of Algorithm 1

The first path to be found is $p_1 = \langle 1,5,7\rangle$, the shortest path from 1 to 7 in G_{11}. As $c_1(p_1) = c^1_1$ and $c_2(p_1) = c^2_2$, p_1 is stored at position C_{12} and S_{12} is set to 0. This means that the shortest path problem in G_{12} does not need to be solved and the next graph to be considered is G_{13}. In this graph the computed path, $p_2 = \langle 1,3,6,4,7\rangle$, is stored in C_{24}, because $c_1(p_2) = c^1_2$ and $c_2(p_2) = c^2_4$, and S_{ij} is set to 0, for $i = 1,2$ and $j = 3,4$. Note that $(3,2)$ is the final position of p_5, and when this path is obtained we have $c_3(p_5) = Q_{32}$, therefore p_5 is not stored in C. In this example, matrix C is unchanged during the second phase.

The following results prove that Algorithm 1 finds a minimal complete set of paths. Theorem 1 shows that after Algorithm 1 is applied all paths in C are Pareto-optimal.

Theorem 1. *For any $C_v = p \neq \emptyset$ after Algorithm 1 we have that p is Pareto-optimal.*

Proof. Path p is the shortest in some graph $G_{\bar{v}}$ such that $\bar{v} \leq v$ and $c(p) = (c^1_{v_1}, c^2_{v_2}, c_3(p))$. Assume,

by contradiction, that there is a path $\hat{p}$ that dominates p, that is,

$$c(\hat{p}) = (c^1_{\hat{v}_1}, c^2_{\hat{v}_2}, c_3(\hat{p})) \leq c(p) = (c^1_{v_1}, c^2_{v_2}, c_3(p)) \text{ and } c(\hat{p}) \neq c(p).$$

Then $\bar{v} \leq v \leq \hat{v}$, therefore $\hat{p}$ is also a path in $G_{\bar{v}}$. On the one hand p is the shortest path in that graph, and on the other hand $c(p) \geq c(\hat{p})$, thus $c_3(\hat{p}) = c_3(p)$. This last equality together with $c(\hat{p}) \leq c(p)$ and $c(\hat{p}) \neq c(p)$ leads to $c^1_{\hat{v}_1} < c^1_{v_1}$ or $c^2_{\hat{v}_2} < c^2_{v_2}$.
Let us consider $c^1_{\hat{v}_1} < c^1_{v_1}$ (the other case can be treated similarly). Then $\hat{v} \geq v$ with $\hat{v}_1 > v_1$, therefore $\hat{p}$ is a path in G_{v^+}, where $v^+ = (v_1 + 1, v_2)$. Two cases can be considered for iteration v^+:

1. If the shortest path problem is solved in G_{v^+}, the solution is p^+ such that $c_3(p^+) \leq c_3(\hat{p}) = c_3(p)$ and p^+ is inserted in $C_{\tilde{v}}$, for some $\tilde{v} \geq v^+$. Furthermore $\tilde{v} \geq v$ with $\tilde{v}_1 \geq v_1 + 1$, thus in the second phase, when $i = v_1 + 1$ and $j = v_2$ we have $Q_{ij} \leq c_3(p^+) \leq c_3(p) = Q_v = Q_{i-1j}$, then C_v is set to $\emptyset$.

2. Else in some iteration v^1 such that $v^1 \leq v^+$, a path p^+ is obtained and inserted in $C_{\tilde{v}}$ for some $\tilde{v} \geq v^+$. We have $\hat{p}$ in G_{v^1} because $v^1 \leq v^+$, then $c_3(p^+) \leq c_3(\hat{p}) = c_3(p)$. Using the same proof we again conclude that $C_v = \emptyset$.

Thus $c^1_{\hat{v}_1} < c^1_{v_1}$ implies $C_v = \emptyset$, which contradicts $C_v = p \neq \emptyset$, therefore p is Pareto-optimal. □

Lemma 1 is an auxiliary result for proving Theorem 2.

Lemma 1. *A Pareto-optimal path is never deleted from C by Algorithm 1.*

Proof. In the first phase a solution p is only replaced by another solution $\bar{p}$, when $\bar{p}$ dominates p. A path p is only deleted in the second phase if p is dominated by another path in C. Therefore a Pareto-optimal path is never removed from C. □

Theorem 2 shows that after Algorithm 1, for each Pareto-optimal path, the matrix C contains a path with the same objective vector.

Theorem 2. *Let p^* be a Pareto-optimal path and v be such that $c_k(p^*) = c^k_{v_k}$, $k = 1, 2$. After the application of Algorithm 1, $C_v = p$, with p some path in $\mathcal{P}$ such that $c(p) = c(p^*)$.*

Proof. Two cases have to be analyzed:

1. The shortest path problem in G_v is not solved. In this case a path p is obtained in $G_{\bar{v}}$, where $\bar{v} \leq v$, and inserted in position $\hat{v}$ such that $\hat{v} \geq v$. Then $c_k(p) = c^k_{\hat{v}_k} \leq c^k_{v_k} = c_k(p^*)$, $k = 1, 2$. Since $\bar{v} \leq v$ the path p^* is in graph $G_{\bar{v}}$, and therefore $c_3(p) \leq c_3(p^*)$, hence $c(p) \leq c(p^*)$. As p^* is Pareto-optimal, then $c(p) = c(p^*)$ and $\hat{v} = v$. This means p is also Pareto-optimal and, at iteration $\bar{v}$, p is stored in C_v.

2. The shortest path problem in G_v is solved. p^* is a path in graph G_v. If p is the shortest path obtained at iteration v, then $c_3(p) \leq c_3(p^*)$. As v is p^*'s final position and $p \in G_v$, then $c_k(p) \leq c_k(p^*)$, $k = 1, 2$, therefore $c(p) \leq c(p^*)$. On the other hand p^* is Pareto-optimal, therefore $c(p) = c(p^*)$ must hold, which implies p is a Pareto-optimal too and v is its final position. Finally $C_v = p$ at the end of iteration v.

In both cases, by Lemma 1 $C_v = p$ holds. □

Therefore the set of paths in C after Algorithm 1 is a minimal complete set of Pareto-optimal solutions. All solutions are Pareto-optimal (Theorem 1) and for any Pareto-optimal we have a path in C with the same objective vector (Theorem 2). Moreover, note that the paths in C have distinct objective vectors. The paths are inserted in the final positions and there is at most one path in each C position, thus there are no two paths with exactly the same values in c_1, c_2.

The number of operations performed by Algorithm 1 depends on the number of distinct values of the bottleneck objective functions, $m_1, m_2 \leq m$, which are correlated with the number of Pareto-optimal solutions, and thus the number of shortest path problems that have to be solved. As a result the new method shares the same worst-case complexity order of previous algorithms in the literature, $\mathcal{O}(m_1 m_2 c(n))$, where $c(n)$ is the number of operations needed to find the shortest path in a network with n nodes. The same bound is valid for the variant that avoids ties in the paths' bottleneck values, as in a worst-case there is a non-dominated solution for every pair of bottleneck values c_1 and c_2.

As for the tricriteria label setting algorithm, each node may have at most $m_1 m_2$ labels, and the binary heap used to manage all temporary labels together can have at most $n m_1 m_2$ elements in total, which has a time complexity of $\mathcal{O}(\log(n m_1 m_2))$. Moreover, in the worst-case, each arc is analyzed $m_1 m_2$ times, $\mathcal{O}(m_1 m_2 \log(n m_1 m_2))$, and for each the dominance test has to be applied, which takes $\mathcal{O}(m m_1^2 m_2^2 \log(n m_1 m_2))$ time. In the worst-case the label setting algorithm performs $\mathcal{O}(m m_1^2 m_2^2 \log(n m_1 m_2) + n m_1 m_2 \log(n m_1 m_2))$ operations.

4 Computational experiments

Computational experiments were carried out to evaluate the performance of the new methods, as well as to compare them with previous approaches. Seven codes were implemented in C: the methods described in the literature, `MMS` and `MMS-R` [14, 15], the stair method, `MMS-S` [17], the blocks method, `MMS-B`, and the label setting algorithm, `MMS-LS`, as well as the variants of `MMS-S` and `MMS-B` which deal with cost ties, `MMS-ST` and `MMS-BT`. The stair method can be seen as a preliminar version of the blocks method, which still intends to use the objective values of the paths computed to avoid solving some shortest path problems, but taking into account only one of the objective functions c_1, c_2. More details about this method can be found in [17]. In order to determine the shortest path, the first four programs used Dijkstra's algorithm [6], implemented with a binary heap to manage the temporary labels. The last two used its modified version described in Section 2. `MMS-LS` was implemented with a binary heap too, that manages all temporary labels together. The codes ran on an Intel Core 2 Duo 2.0GHz with 3GB of RAM.

Random networks with 2 000, 7 000, 12 000, 15 000, 20 000 and 30 000 nodes, dn arcs, for densities $d = 5, 20$ and 100, and integer c_{ij}^3 uniformly generated in $[1, M]$, with $M = 10, 50, 1\,000$, were considered. The bottleneck values c_{ij}^k are integers randomly generated in $[1, m_k]$, $k = 1, 2$, where $m_1 = m_2$ may be 20, 100, 500 and 1 000. For any node i, d successors j are randomly chosen and the arcs (i, j) are created in $\mathcal{A}$. The networks were represented using the forward star form [1].

All the results are average values for 10 different instances of each data set dimension. Tables 2,

3 and 4 present the number of computed Pareto-optimal solutions, the number of solved shortest path problems and the CPU times (in seconds).

					MMS	MMS-R		MMS-S		MMS-B		MMS-LS	
n	m	m_1	c_{ij}^3	#PO	T(s)	#SPP	T(s)	#SPP	T(s)	#SPP	T(s)	#Labels	T(s)
2 000	9 983	20	50	36	0.1	135	0.1	58	0.0	47	0.0	67 905	0.3
2 000	39 791	20	50	88	0.3	311	0.3	150	0.2	113	0.1	159 811	2.3
2 000	9 984	100	50	42	2.0	2 205	0.8	104	0.1	63	0.0	142 187	0.7
2 000	39 790	100	50	206	7.6	7 088	7.2	624	0.7	299	0.4	398 303	8.4
2 000	9 984	20	1 000	30	0.1	148	0.1	51	0.0	40	0.0	71 920	0.3
2 000	39 791	20	1 000	82	0.3	307	0.3	127	0.1	101	0.1	172 692	2.6
2 000	9 984	100	1 000	65	2.6	2 505	1.6	156	0.1	91	0.1	153 874	0.8
2 000	39 795	100	1 000	195	8.0	6 813	7.3	562	0.7	275	0.4	411 252	9.0
7 000	34 983	20	50	35	0.4	119	0.3	52	0.1	43	0.1	294 238	2.0
7 000	139 782	20	50	84	1.3	282	1.2	139	0.7	107	0.6	642 390	13.6
7 000	34 987	100	50	49	7.6	1 993	3.6	114	0.3	70	0.2	653 195	5.8
7 000	139 778	100	50	246	28.0	6 920	24.7	705	3.0	352	1.6	1 778 927	53.4
7 000	34 984	20	1 000	43	0.3	147	0.2	67	0.1	55	0.1	317 447	2.1
7 000	139 788	20	1 000	94	1.3	298	1.3	148	0.7	114	0.6	679 620	14.4
7 000	34 984	100	1 000	71	10.2	1 996	5.5	157	0.5	98	0.3	699 181	6.2
7 000	139 793	100	1 000	247	33.6	6 535	30.8	736	3.8	344	1.9	1 738 864	55.5
12 000	59 983	20	50	41	0.8	114	0.6	61	0.3	52	0.3	548 579	4.3
12 000	239 785	20	50	90	2.8	269	2.7	143	1.6	113	1.3	1 087 422	25.2
12 000	59 984	100	50	86	14.2	2 325	7.0	219	0.9	123	0.5	1 322 491	13.7
12 000	239 794	100	50	246	56.6	6 664	47.9	722	5.9	347	3.0	3 419 712	115.1
12 000	59 985	20	1 000	47	0.8	132	0.5	68	0.3	58	0.3	594 658	4.7
12 000	239 783	20	1 000	103	2.6	289	2.4	154	1.5	122	1.3	1 252 525	30.4
12 000	59 984	100	1 000	99	17.7	2 521	10.6	241	1.2	138	0.7	1 447 591	15.4
12 000	239 782	100	1 000	281	62.8	6 678	55.9	804	7.9	384	4.0	3 475 595	126.2
15 000	74 987	20	50	45	0.7	136	0.5	64	0.3	55	0.3	686 606	5.7
15 000	299 784	20	50	88	3.2	285	2.9	140	1.7	110	1.4	1 470 148	35.4
15 000	74 987	100	50	134	18.9	3 219	14.9	337	1.8	191	1.0	1 825 650	20.3
15 000	299 781	100	50	301	70.8	7 036	64.5	899	9.3	424	4.7	4 425 397	156.3
15 000	74 985	20	1 000	58	0.9	151	0.8	82	0.5	70	0.4	724 283	6.2
15 000	299 784	20	1 000	97	3.5	285	3.0	151	1.9	117	1.6	1 650 018	42.9
15 000	74 985	100	1 000	89	20.5	2 463	11.0	229	1.2	126	0.8	1 819 763	20.8
15 000	299 779	100	1 000	299	76.1	6 868	68.4	873	9.8	412	4.9	4 498 181	168.0

Table 2: Average results for codes MMS, MMS-R, MMS-S, MMS-B and MMS-LS.

The first set of tests aimed to compare MMS-B and MMS-LS against the previous methods, MMS, MMS-R and MMS-S. We first note that the number of Pareto-optimal solutions for this set of problems is far from the theoretical upper bound $m_1 m_2$ [14]. For $m_1 = m_2 = 20$ that number is between 7.5% and 25.8% of $m_1 m_2$ and for $m_1 = m_2 = 100$ between 0.4% and 3.0% of $m_1 m_2$. As for the number of shortest path problems solved, it is worth mentioning that the $m_1 m_2$ problems, that are always solved by MMS, was reduced in more recent approaches. For MMS-B this number varied between $10.0\% m_1 m_2$ and $30.5\% m_1 m_2$ when $m_1 = m_2 = 20$. The numbers decrease when m_1, m_2 increase and are between 0.6% and 4.2%, when $m_1 = m_2 = 100$. For both cases the ratios are higher if the networks are denser. It should be noted that the number of subproblems solved by MMS-B and the number of Pareto-optimal solutions are very close (the first is never greater than 1.5 times the second), which indicates that not many unnecessary shortest paths are being computed.

With respect to MMS-LS Table 2 shows that the number of labels examined can be very high.

Consequently this algorithm was not as efficient as the others. Moreover, we note that the performance of this method is highly dependent on the number of arcs, m. Finally, it is worth mentioning that `MMS-B` was the fastest code for all the instances reported in Table 2.

On this set of instances the codes `MMS-ST` and `MMS-BT` were always less efficient than their original versions. In fact, the number of shortest path problems solved coincided and, as expected, the running times were higher for the codes that deal with ties on the paths cost. On a second test set tighter ranges of the c^3_{ij} values, $(i,j) \in \mathcal{A}$, were considered. This set was formed by random networks with 7 000 and 15 000 nodes, densities of $d = 5, 20, 100$, and c^3_{ij} uniformly generated in $[1, 10]$. Two numbers of bottleneck values were considered, $m_1 = m_2 = 20, 100$. The results, summarized in Table 3, are again averages for 10 problems.

				`MMS-S`		`MMS-ST`		`MMS-B`		`MMS-BT`	
n	m	m_1	#PO	#SPP	T (s)	#SPP	T (s)	#SPP	T (s)	#SPP	T (s)
7 000	34 982	20	40	62	0.1	60	0.1	54	0.1	51	0.1
7 000	139 790	20	67	120	0.5	109	0.5	93	0.4	84	0.4
7 000	694 933	20	76	150	2.2	115	1.8	111	1.7	90	1.5
7 000	34 986	100	106	261	0.5	246	0.5	158	0.3	151	0.3
7 000	139 791	100	186	560	2.7	500	2.5	294	1.4	266	1.4
7 000	694 907	100	265	892	12.7	706	10.2	435	6.5	365	5.6
15 000	74 984	20	47	74	0.3	72	0.3	62	0.3	60	0.3
15 000	299 784	20	84	145	1.8	134	1.8	112	1.4	102	1.4
15 000	1 494 822	20	88	169	5.7	133	4.8	129	4.6	104	4.0
15 000	74 985	100	116	291	1.4	278	1.4	169	0.8	163	0.8
15 000	299 787	100	234	715	7.5	627	6.8	352	3.7	322	3.6
15 000	1 494 774	100	303	1 030	32.0	806	25.6	489	16.0	408	13.8

Table 3: Average results for codes `MMS-S`, `MMS-ST`, `MMS-B` and `MMS-BT`, in instances with $c^3_{ij} \in [1, 10]$.

Table 3 shows that for costs c^3_{ij} in $[1, 10]$ avoiding ties may have a positive impact over both the number of shortest path algorithm calls and the running times. The general tendency of increasing CPU times and number of subproblems with n and, in particular, with m_1 and m_2 is still observed for `MMS-ST` and `MMS-BT`. This modification has more impact on the stair method than on the blocks method. The improvement occurred only in problems with $c^3_{ij} \in [1, 10]$. When considering wider intervals for the values c^3_{ij}, `MMS-ST` and `MMS-BT` were always worse than `MMS-S` and `MMS-B`, respectively.

The last set of results shows that bigger problems can still be solved by the blocks algorithm in few seconds. The number of nodes considered was 7 000, 15 000, 20 000 and 30 000, the densities $d = 5, 20$, and $c^3_{ij} \in [1, 1\,000]$. Two numbers of bottleneck values were considered, $m_1 = m_2 = 500, 1\,000$.

Concerning Table 4 it should be stressed that the cardinality of the minimal complete set is still small, even though greater values of m_k are considered, $k = 1, 2$. Besides, on the one hand the number of computed Pareto-optimal solutions was between 0.01% and 0.2% of $m_1 m_2$, and on the other the number of shortest path problems solved was, as before, very close. This indicates that both factors are highly correlated with the graph's density, and thus with the number of arcs, m.

				MMS-B	
n	m	m_1	#PO	#SPP	T (s)
7 000	34 986	500	121	179	0.5
7 000	139 796	500	374	564	3.1
15 000	74 984	500	199	297	1.7
15 000	299 789	500	434	666	7.9
20 000	99 985	500	129	195	1.6
20 000	399 770	500	411	622	9.8
20 000	100 000	1 000	129	196	1.6
20 000	400 000	1 000	497	773	11.8
30 000	150 000	1 000	238	364	4.4
30 000	600 000	1 000	538	837	19.9

Table 4: Average results for code MMS-B in instances with $c^3_{ij} \in [1, 1\,000]$.

5 Final remarks

We have introduced a method for finding the minimal complete set of Pareto-optimal paths for a tricriteria path problem involving two bottleneck objective functions and a cost, the blocks method. This method enhances the procedures presented recently in [14, 15, 17] by using the objective values of the obtained paths to reduce the number of shortest path subproblems that have to be solved. As a result the new method has the same worst-case complexity order as previous algorithms, $\mathcal{O}(m_1 m_2 c(n))$, where $c(n)$ is the number of operations needed to find the shortest path in a network with n nodes. Empirical tests showed that in practice this bound is far from being attained and in random instances with $n = 15\,000$, average degree 20 and $m_1 = m_2 = 100$ the blocks method improved the previous solving about 16 times less shortest path problems than before. As a result the CPU times were improved in about 14 times. The blocks method was able to compute the minimal complete set in instances of $n = 30\,000$ and $m_1 = m_2 = 1000$ in less than 20 seconds.

A variant of the method with the purpose of reducing the number of subproblems that have to be solved was also proposed. To this end the labeling test used in the shortest path algorithm is modified so that every time two paths ending at a given node have the same cost, the one with the best bottleneck value is chosen. For the experiments performed on random instances the percentage of improvement with this variant was at most 15% for arc costs in $[1, 10]$. A label setting algorithm, with worst-case complexity order of $\mathcal{O}(m m_1^2 m_2^2 \log(n m_1 m_2))$ was also proposed, however the high number of labels it has to manage led it to a poor performance in the tests performed, when compared to the others.

Finally, it should be noted that the idea used to develop the blocks method defines a way to search for the best solutions in matrix C. Therefore it can be applied to other combinatorial problems with one additive and two bottleneck functions. Moreover this procedure can be extended to the multi-objective bottleneck network problem, similarly to what was done in [16].

Acknowledgments This work was developed during a visit of L. Pinto and M. Pascoal to CIRRELT, Montréal, Canada. It was partly supported by Government of Canada, Graduate Students Exchange Program, by the National Council for Scientific and Technological Development of Brazil (CNPq), and by the FCT Portuguese Foundation of Science and Technology (Fundação para a

Ciência e a Tecnologia) under grant SFRH/BSAB/830/2008 and project POSC/U308/1.3/NRE/04. The authors also thank Gilbert Laporte for comments and suggestions on a preliminary version of this manuscript.

References

[1] Ahuja, R., Magnanti, T., Orlin, J. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] Berman, O., Einav, D., Handler, G. "The constrained bottleneck problem in network". *Operations Research* 38, pp. 178-181, 1990.

[3] Brumbaugh-Smith, J. and Shier, D. R. "An empirical investigation of some bicriterion shortest path algorithms". *European Journal of Operational Research* 43, pp. 216-224, 1989.

[4] Clímaco, J., Martins, E. "On the determination of the nondominated paths in a multiobjective network problem". Proc. of the V Sympösium über Operations Research, Köln, 1980, in *Methods in Operations Research*, Anton Hain, Königstein 40, pp. 255-258, 1981.

[5] Clímaco, J., Martins, E. "A bicriterion shortest path algorithm". *European Journal of Operational Research* 11, pp. 399-404, 1982.

[6] Dijkstra, E. "A note on two problems in connection with graphs". *Numerical Mathematics* 1, pp. 395-412, 1959.

[7] Gandibleux, X., Beugnies, F., Randriamasy. S. "Multi-objective shortest path problems with a maxmin cost function". *4OR – Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 4, pp. 47-59, 2006.

[8] Guerriero, F., Musmanno, R., "Label correcting methods to solve multicriteria shortest path problems". *Journal of Optimization Theory and Applications* 111, pp. 589-613, 2001.

[9] Hansen, P. "Bicriterion path problems". In: *Multicriteria decision making: theory and applications*, Lecture Notes in Economics and Mathematical Systems 177, G. Fandel and T. Gal (eds.), Springer, Heidelberg, pp. 109-127, 1980.

[10] Martins, E. "On a multicriteria shortest path problem". *European Journal of Operational Research* 16, pp. 236-245, 1984.

[11] Martins, E. "On a special class of bicriterion path problems". *European Journal of Operational Research* 17, pp. 85-94, 1984.

[12] Martins, E., Pascoal, M., Rasteiro, D., Santos, J. "The optimal path problem". *Investigação Operacional* 19, pp. 43-60, 1999. (`http://www.mat.uc.pt/~marta/Publicacoes/opath.ps.gz`).

[13] Mote, J., Murthy, I., Olson, D. "A parametric approach to solving bicriterion shortest path problems". *European Journal of Operational Research* 53, pp. 81-92, 1991.

[14] Pinto, L., Bornstein, C., Maculan, N. "The tricriterion shortest path problem with at least two bottleneck objective functions". *European Journal of Operational Research* 198, pp. 387-391, 2009.

[15] Pinto, L., Bornstein, C., Maculan, N. "Um problema de caminho tri-objetivo". *Anais do XL SBPO*, João Pessoa-PB, pp. 1032-1042, 2008. (`http://www.cos.ufrj.br/~leizer/MMS-R.pdf`)

[16] Pinto, L., Bornstein, C., Maculan, N. "A polynomial algorithm for the multi-objective bottleneck network problem with one MinSum objective function". Submitted to *Networks*, 2008.

[17] Pinto, L., Pascoal, M. "Enhanced algorithms for tricriteria shortest path problems with two bottleneck objective functions". Technical Report nº 3, INESC-Coimbra, 2009. (`http://www.inescc.pt/documentos/3_2009.pdf`)

[18] Raith, A., Ehrgott, M. "A comparison of solution strategies for biobjective shortest path problems". *Computers and Operations Research* 36, pp. 1299-1331, 2009.

[19] Skriver, A., Andersen, K. A. "A label correcting approach for solving bicriterion shortest-path problems". *Computers and Operations Research* 27, pp. 507-524, 2000.

[20] Tung, C., Chew, K. "A multicriteria Pareto-optimal path algorithm". *European Journal of Operational Research* 62, pp. 203-209, 1992.

Anexo 3

Este anexo contém o artigo de Pinto e Laporte intitulado *An efficient algorithm for the Steiner tree problem with revenue, bottleneck and hop objective functions*, o qual foi submetido à *European Journal of Operational Research*. A versão inicial foi submetida em abril de 2009. Em outubro recebemos as sugestões dos revisores e as implementamos na atual versão. Este anexo contém essa nova versão, submetida em novembro de 2009.

.

An efficient algorithm for the Steiner tree problem with revenue, bottleneck and hop objective functions

Leizer Lima Pinto[1,2] and Gilbert Laporte[2,3]

[1]COPPE/UFRJ - Federal University of Rio de Janeiro, Brazil
Department of Systems Engineering and Computer Science
E-mail: leizer@cos.ufrj.br

[2]Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
E-mail: Gilbert.Laporte@cirrelt.ca

[3]Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

April 2009
Revised November 2009

Abstract

This paper considers a tricriteria Steiner tree problem arising in the design of telecommunication networks. The objective functions consist of maximizing the revenue and of minimizing the maximal distance between each pair of interconnected nodes, as well as the maximal number of arcs between the root and each node. A polynomial algorithm is developed for the generation of a minimal complete set of Pareto-optimal Steiner trees. Optimality proofs are given and computational experience on a set of randomly generated problems is reported.

Keywords: Steiner tree; Multi-objective problem; Pareto-optimal solution; Bottleneck function.

1 The problem

The classical Steiner Tree Problem (STP) consists of finding a Steiner tree minimizing the sum of its arc costs. This single-objective problem is a well known NP-hard problem in combinatorial optimization (see [10, 12]). In the Prize-Collecting Steiner Tree Problem (PCSTP) the goal is to find a Steiner tree maximizing the difference between the collected revenue, equal to the sum of the node revenues, and the network cost, equal to the sum of the arc costs. The STP is a paticular case of the PCSTP in which all the revenues are zero, and therefore the PCSTP is also NP-hard. The PCSTP has been studied in [4, 8, 13, 14].

A variant of the PCSTP which consists of finding a Steiner tree maximizing the revenue under budget and hop constraints has been studied in Costa *et al.* [7]. The budget is the network cost, i.e., the sum of the arc costs. This problem is NP-hard because of the budget constraint. However, changing this constraint to a bottleneck constraint yields a polynomial problem. Here, the bottleneck and the hop are considered as objective functions, instead of constraints. Single-objective bottleneck Steiner tree problems are considered in [5, 16].

The purpose of this paper is to propose a polynomial algorithm for the generation of a minimal complete set of Pareto-optimal solutions for a tricriteria Steiner tree problem arising in the design of telecommunication networks. The problem is defined on a directed graph $G = (\mathcal{N}, \mathcal{A})$ with $|\mathcal{N}| = n$ and $|\mathcal{A}| = m$. Each arc $(i, j) \in \mathcal{A}$ has an associated length $d_{ij} \in \mathbb{R}$ and each node $i \in \mathcal{N}$ has an associated revenue $r_i \geq 0$. Given a set of *terminal* nodes $\mathcal{T} \subseteq \mathcal{N}$, a directed tree $\bar{G} = (\mathcal{V}, \mathcal{E})$

of G with *root* $s \in \mathcal{T}$ is called a *Steiner tree* when $\mathcal{V} \supseteq \mathcal{T}$. For each $i \in \mathcal{V}$ let $h(i)$ be the number of hops between the root s and i in $\bar{G}$. Defining $\mathcal{S}$ as the set of all Steiner trees of G, the *objective vector* associated with Steiner tree $\bar{G} \in \mathcal{S}$ is given by $f(\bar{G}) = \big(f_1(\bar{G}), f_2(\bar{G}), f_3(\bar{G})\big) \in \mathbb{R}^3$, where

$$f_1(\bar{G}) = \max_{i \in \mathcal{V}} \{h(i)\}, \quad f_2(\bar{G}) = \max_{(i,j) \in \mathcal{E}} \{d_{ij}\} \text{ and } f_3(\bar{G}) = \sum_{i \in \mathcal{V}} r_i.$$

The tricriteria Steiner tree problem with revenue, hop and bottleneck objective functions is then defined as:

$$\begin{array}{ll} \text{minimize} & f_k(\bar{G}), \quad k = 1, 2 \\ \text{maximize} & f_3(\bar{G}) \\ \text{subject to} & \bar{G} \in \mathcal{S}. \end{array}$$

Applications of this problem arise in the design of telecommunications local access networks [4, 6, 8, 13], such as the construction of a fiber-optic network to provide broadband Internet connections to businesses and residential buildings. Here the graph corresponds to a street map since local access fiber is typically laid along streets. The nodes of the graph correspond to the street intersections and to the locations of buildings. To each node representing a building is associated a potential revenue generated by providing telecommunication services to that building. The other nodes have no associated revenue. The cost of creating or maintaining an arc depends on its length. An arc with a large weight in the solution can cause difficulties with respect to network support. Since telecommunications applications usually consider a probability of service failure (here we assume the same probability for all arcs, as in [7, 11]), then a node with a large number of hops between it and the root is more likely to experience failures. Therefore one may be interested in simultaneously maximizing the revenue, as well as minimizing the maximal arc length and the maximal number of hops between the root and any node. Zelikovsky [16] describes several applications in which a bottleneck objective is appropriate. One is a location problem in which hospitals are assigned to potential users by means of arcs. Another application arises in electronic physical design automation where nets are routed subject to delay minimization. In both cases minimizing a bottleneck objective helps maintain service quality by ensuring that transmission or travel times remain acceptable in the worst case.

The remainder of this paper is organized as follows. The algorithm is described in Section 2, followed by an example in Section 3. An optimality proof is provided in Section 4 and computational experiments are reported in Section 5.

2 Algorithm

Graph $\bar{G} \in \mathcal{S}$ *dominates* $\hat{G} \in \mathcal{S}$ whenever $f_k(\bar{G}) \leq f_k(\hat{G})$, $k = 1, 2$, and $f_3(\bar{G}) \geq f_3(\hat{G})$ with $f(\bar{G}) \neq f(\hat{G})$. A Steiner tree $\bar{G} \in \mathcal{S}$ is *Pareto-optimal* if it is not dominated by any other Steiner tree in $\mathcal{S}$. A set $\mathcal{S}^* \subseteq \mathcal{S}$ of Pareto-optimal solutions is a *minimal complete set* if for each $G^1, G^2 \in \mathcal{S}^*$ we have $f(G^1) \neq f(G^2)$ and for any Pareto-optimal solution $G^* \in \mathcal{S}$ there exists $\bar{G} \in \mathcal{S}^*$ so that $f(\bar{G}) = f(G^*)$.

Let m_1 be the number of different values of $d_{ij}, (i,j) \in \mathcal{A}$, and suppose they are decreasing order, i.e., $d_1 > d_2 > \cdots > d_{m_1}$. Considering $\mathcal{A}_k = \{(i,j) \in \mathcal{A} \mid d_{ij} \leq d_k\}$, the subgraph of G, $G_k = (\mathcal{N}, \mathcal{A}_k)$ can be defined. It is easy to see that $G_1 = G$.

We first describe Algorithm 1 which is an auxiliary procedure used by our main algorithm. It seeks to determine an optimal Steiner tree $\bar{G} = (\mathcal{V}, \mathcal{E})$, with respect to the revenue objective function, in a subgraph G_i of G with no more than h hops between the root s and any node. The

algorithm uses a breadth-first search until it finds all nodes j for which there exists a path with at most h hops from s to j in G_i. The optimality of the algorithm is guaranted because the revenues are non-negative. Some auxiliary structure are used, namely $\mathcal{F}, q, r$ and H_k, where $\mathcal{F}$ is the set of scanned nodes, i.e., $j \in \mathcal{F}$ means that $l \in \mathcal{V}, \forall (j,l) \in \mathcal{A}_i$; q is the node being scanned at the current iteration; r is the sum of the revenues of the nodes in $\mathcal{V}$; and H_k contains the number of arcs from s to k in $\bar{G}$. A similar algorithm that spans all nodes, i.e., that finds a spanning tree of G using a breadth-first search, is presented in Gabow and Tarjan [9].

Algorithm 1. *Finding a Steiner tree in* G_i
`Given:` G_i, h, $\mathcal{T}$ and s
`Do:` $\mathcal{V} \leftarrow \{s\}$, $\mathcal{E} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$, $q \leftarrow s$, $r \leftarrow r_s$, $H_s \leftarrow 0$ and $H_k \leftarrow \infty, \forall k \in \mathcal{N} - \{s\}$
`While` $H_q < h$ and $\mathcal{F} \neq \mathcal{V} \neq \mathcal{N}$ `Do`
  `For all` $(q,l) \in \mathcal{A}_i$ with $H_l = \infty$ `Do`
    $H_l \leftarrow H_q + 1, \quad r \leftarrow r + r_l, \quad \mathcal{V} \leftarrow \mathcal{V} \cup \{l\}$ and $\mathcal{E} \leftarrow \mathcal{E} \cup \{(q,l)\}$
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{q\}$
  $q \leftarrow j \in \mathcal{V} - \mathcal{F}$ such that $H_j \leq H_k, \forall k \in \mathcal{V} - \mathcal{F}$
`If` $\mathcal{T} \subseteq \mathcal{V}$ `Then Return` $(\mathcal{V}, \mathcal{E})$
`Else Return` $\emptyset$

The complexity of Algorithm 1 is $\mathcal{O}(n^2)$. In the *while* loop there are never more than n values of q. Of course, for any q the maximum number of nodes l so that $(q,l) \in \mathcal{A}_i$ is $n-1$. Moreover, the choice of q at each iteration of the loop is made in $\mathcal{V} - \mathcal{F}$ operations, where $|\mathcal{V} - \mathcal{F}| \leq n$.

Let S_i^h be the set of Steiner trees, with $f_1 \leq h$, in G_i. The following lemma proves the optimality of Algorithm 1.

Lemma 1. *Let* $\bar{G} = (\mathcal{V}, \mathcal{E})$ *be the solution obtained by Algorithm 1 applied to* G_i *for a given* h. *Then* $f_3(\bar{G}) \geq f_3(\hat{G}), \forall \hat{G} = (\hat{\mathcal{V}}, \hat{\mathcal{E}}) \in S_i^h$.

Proof. After Algorithm 1, $\mathcal{V}$ is the set of all nodes $j \in \mathcal{N}$ such that there exists a path from s to j in G_i with no more than h intermediate arcs. Then $\mathcal{T} \not\subseteq \mathcal{V} \Leftrightarrow S_i^h = \emptyset$, and when $S_i^h \neq \emptyset$ we have $\hat{\mathcal{V}} \subseteq \mathcal{V}, \forall \hat{G} = (\hat{\mathcal{V}}, \hat{\mathcal{E}}) \in S_i^h$, which implies $f_3(\bar{G}) \geq f_3(\hat{G})$, because $r_k \geq 0, \forall k \in \mathcal{N}$. Therefore $\bar{G}$ is the best solution in G_i with respect to the revenue function and with $f_1 \leq h$. □

We now present an algorithm for finding a minimal complete set of Pareto-optimal solution. It contains two phases: Pareto-optimal candidates are found in the first phase and the second phase eliminates dominated and equivalent solutions. Here C and Q are two $(n-1) \times m_1$ matrices used to store the solutions obtained by algorithm and their f_3 objective values, respectively. They allow the determination of S^* and its objective vectors. A binary matrix R of dimension $(n-1) \times m_1$ marks the iterations (subproblems) that have to be executed. Algorithm 1 is called at iteration $v = (v_1, v_2)$ of Algorithm 2 if and only if $R_v = 1$; otherwise the next iteration is considered. Variable *ctrl* is used to skip subproblems that have no solution.

The first phase consists of two loops. The indices associated with the outer v_1-*loop* and the inner v_2-*loop* define, respectively, which h and which subgraph, G_{v_2}, should to be considered by Algorithm 1 at iteration $v = (v_1, v_2)$. If a solution $\bar{G}$ is obtained at iteration v, then it is inserted in $C_{\bar{v}}$, where $\bar{v}$ is such that $f_1(\bar{G}) = n - \bar{v}_1$ and $f_2(\bar{G}) = d_{\bar{v}_2}$. The value $f_3(\bar{G})$ is inserted in $Q_{\bar{v}}$, and $R_{\hat{v}}$ is set to 0 for every $\hat{v}$ such that $v \leq \hat{v} \leq \bar{v}$. In the second phase the matrix Q is used to compare the solutions obtained during the first phase and then, the dominated and equivalent solutions are eliminated from matrix C.

Algorithm 2 is similar to the Blocks algorithm presented by Pinto and Pascoal [15] for tricriteria shortest path problems.

Algorithm 2. *Finding a minimal complete set of Pareto-optimal Steiner trees*

`For` $i = 1, \ldots, n-1$ `Do`
  `For` $j = 1, \ldots, m_1$ `Do` $C_{ij} \leftarrow \emptyset$, $Q_{ij} \leftarrow -\infty$ and $R_{ij} \leftarrow 1$
$ctrl \leftarrow m_1$ and $v_1 \leftarrow 1$
Phase 1. *Finding Pareto-optimal candidates*
`While` $v_1 \leq n-1$ and $ctrl \neq 0$ `Do`
  $v_2 \leftarrow 1$
  `While` $v_2 \leq ctrl$ `Do`
    `If` $R_v = 1$ `Then`
      $\bar{G} \leftarrow$ the result of Algorithm 1 for G_{v_2}, $h = n - v_1$, $\mathcal{T}$ and s
      `If` $\bar{G} = \emptyset$ `Then` $ctrl \leftarrow v_2 - 1$
      `Else`
        $\bar{v}_1 \leftarrow n - f_1(\bar{G})$ and $\bar{v}_2 \leftarrow g$ such that $f_2(\bar{G}) = d_g$
        `If` $f_3(\bar{G}) > Q_{\bar{v}}$ `Then`
          $C_{\bar{v}} \leftarrow \bar{G}$ and $Q_{\bar{v}} \leftarrow f_3(\bar{G})$
          `For` $i = v_1, \ldots, \bar{v}_1$ `Do`
            `For` $j = v_2, \ldots, \bar{v}_2$ `Do` $R_{ij} \leftarrow 0$
    $v_2 \leftarrow v_2 + 1$
  $v_1 \leftarrow v_1 + 1$
Phase 2. *Filtering Pareto-optimal solutions in C*
`For` $i = n-1, \ldots, 1$ `Do`
  `For` $j = m_1, \ldots, 1$ `Do`
    If $Q_{ij} \neq -\infty$ `Then`
      If $j > 1$ and $Q_{ij} \geq Q_{ij-1}$ `Then` $C_{ij-1} \leftarrow \emptyset$, $Q_{ij-1} \leftarrow Q_{ij}$
      If $i > 1$ and $Q_{ij} \geq Q_{i-1j}$ `Then` $C_{i-1j} \leftarrow \emptyset$, $Q_{i-1j} \leftarrow Q_{ij}$

The complexity of Algorithm 2 is $\mathcal{O}(m_1 n^3)$. The number of iteration in ***Phase 1*** is no more than $(n-1)m_1$. The complexity of each iteration is $\mathcal{O}(n^2)$, which is the complexity of Algorithm 1. Thus the complexity of ***Phase 1*** is $\mathcal{O}(m_1 n^3)$. It is easy to see that the complexity of ***Phase 2*** is $\mathcal{O}(m_1 n)$.

Naturally, we always have $v \leq \bar{v}$ in Algorithm 2, where $\bar{v}$ is the *insertion position*, in C, of the Steiner tree $\bar{G}$ obtained at iteration v. By Algorithm 1 $f_1(\bar{G}) \leq h = n - v_1$, implying that $v_1 \leq n - f_1(\bar{G}) = \bar{v}_1$. Since $\bar{G} = (\mathcal{V}, \mathcal{E})$ was obtained in $G_{v_2} = (\mathcal{N}, \mathcal{A}_{v_2})$ then $\mathcal{E} \subseteq \mathcal{A}_{v_2}$ leading to $d_{\bar{v}_2} = f_2(\bar{G}) \leq d_{v_2}$, which implies that $v_2 \leq \bar{v}_2$.

3 Example

To illustrate Algorithm 2, we apply it to graph G depicted in Figure 1. For this instance $m_1 = 5$ with $d_{ij} \in \{9, 8, 4, 3, 1\}$, for any $(i, j) \in \mathcal{A}$.

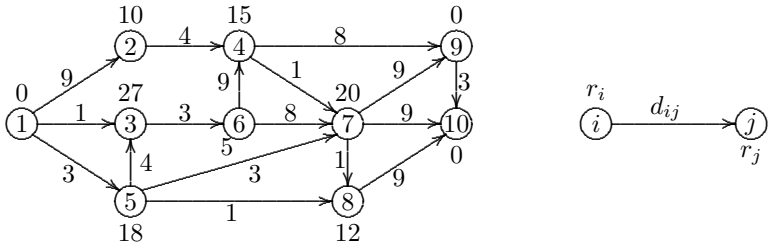
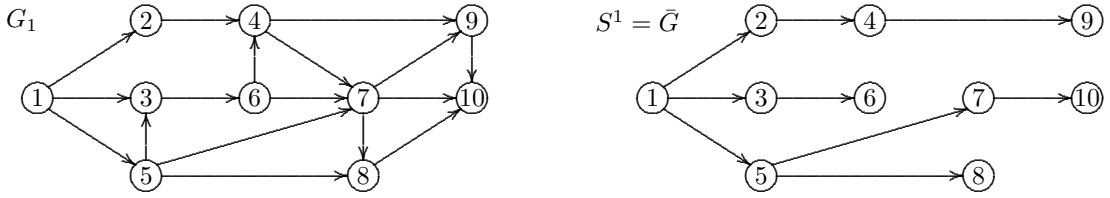


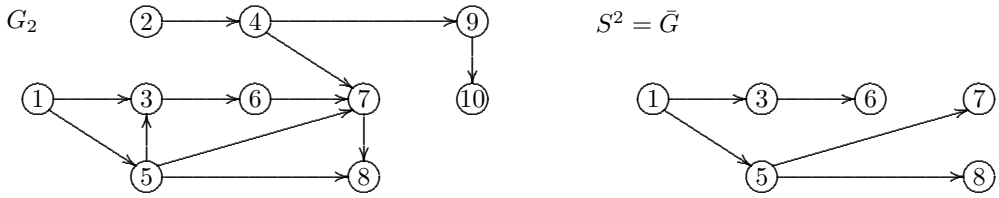
Figure 1: Graph G.

We consider $\mathcal{T} = \{s = 1, 3, 7, 8\}$. Figures 2, 3 and 4 depict the solutions obtained in *Phase 1*. These figures contain the subgraph G_{v_2} at iteration v, the Steiner tree obtained by Algorithm 1 in G_{v_2} and the updates made at the end of these iterations.



$\bar{v}_1 \leftarrow 10 - 3$, $\bar{v}_2 \leftarrow 1$, $C_{71} \leftarrow S^1$, $Q_{71} \leftarrow 107$ and $R_{ij} \leftarrow 0$, for $1 \leq i \leq 7$ and $j = 1$.

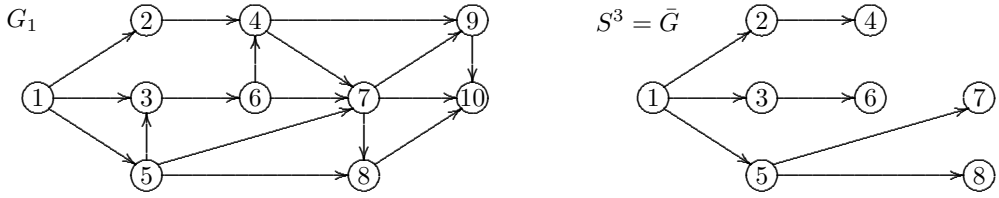
Figure 2: First iteration, $h = 9$ and $v = (1, 1)$.



$\bar{v}_1 \leftarrow 10 - 2$, $\bar{v}_2 \leftarrow 4$, $C_{84} \leftarrow S^2$, $Q_{84} \leftarrow 82$ and $R_{ij} \leftarrow 0$, for $1 \leq i \leq 8$ and $2 \leq j \leq 4$.

Figure 3: Second iteration, $h = 9$ and $v = (1, 2)$.

At iterations three, $v = (1, 3)$, and four, $v = (1, 4)$, we have $R_v = 0$. The next iteration with $R_v = 1$ is $v = (1, 5)$, where $\mathcal{A}_5 = \{(1, 3), (4, 7), (5, 8), (7, 8)\} \Rightarrow \bar{G} = \emptyset \Rightarrow ctrl \leftarrow v_2 - 1 = 4$. This, together with the updates made at iterations $(1, 1)$ and $(1, 2)$, implies that the next iteration with $v_2 \leq ctrl$ and $R_v = 1$ is $v = (8, 1)$.



$\bar{v}_1 \leftarrow 10 - 2$, $\bar{v}_2 \leftarrow 1$, $C_{81} \leftarrow S^3$, $Q_{81} \leftarrow 107$ and $R_{81} \leftarrow 0$.

Figure 4: Iteration $v = (8, 1)$ and $h = 2$.

At iterations $v = (8, 2)$, $(8, 3)$ and $(8, 4)$ we have $R_v = 0$. At iteration $v = (9, 1)$, since $h = 1$ then $\bar{G} = \emptyset \Rightarrow ctrl \leftarrow v_2 - 1 = 0$. This leads at the end of the *Phase 1*. Figure 5 shows the matrix C after *Phase 2*, i.e., the final result of Algorithm 2.

4 Optimality proof

We now provide the optimality proof of Algorithm 2, i.e., we prove that it can find a minimal complete set of Pareto-optimal solutions. We will first show that all Steiner trees in C are Pareto-

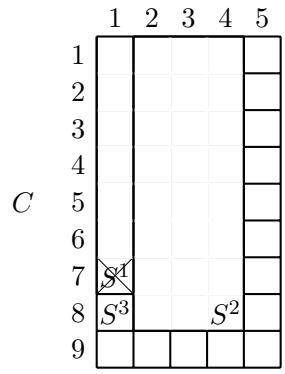


Figure 5: Matrix C after *Phase 2* of Algorithm 2.

optimal solutions after the execution of Algorithm 2.

Theorem 1. *If $C_{\bar{v}} = \bar{G} \neq \emptyset$ after Algorithm 2, for any $\bar{v}$, then $\bar{G}$ is a Pareto-optimal solution.*

Proof. By Algorithm 2 $f_1(\bar{G}) = n - \bar{v}_1$ and $f_2(\bar{G}) = d_{\bar{v}_2}$. The proof is by contradiction. Suppose that $\bar{G}$ is not a Pareto-optimal solution. This implies that $\bar{G}$ is dominated by some $\hat{G} \in \mathcal{S}$, i.e., $f_k(\hat{G}) \leq f_k(\bar{G})$, $k = 1, 2$, and $f_3(\hat{G}) \geq f_3(\bar{G})$, with $f(\hat{G}) \neq f(\bar{G})$. We have $f_1(\hat{G}) = n - \hat{v}_1$ and $f_2(\hat{G}) = d_{\hat{v}_2}$ for some $\hat{v}$ so that $1 \leq \hat{v}_1 \leq n-1$ and $1 \leq \hat{v}_2 \leq m_1$. Then $n - \hat{v}_1 = f_1(\hat{G}) \leq f_1(\bar{G}) = n - \bar{v}_1 \Rightarrow \hat{v}_1 \geq \bar{v}_1$ and $d_{\hat{v}_2} = f_2(\hat{G}) \leq f_2(\bar{G}) = d_{\bar{v}_2} \Rightarrow \hat{v}_2 \geq \bar{v}_2$, so $\hat{v} \geq \bar{v}$. It can be seen that $f_2(\hat{G}) = d_{\hat{v}_2}$ ensures that $\hat{G}$ is in G_g, for all $g \leq \hat{v}_2$. By the analysis presented after the description of Algorithm 2, $\bar{v} \geq v$, where v is the iteration at which $\bar{G}$ was obtained. Therefore $\hat{v} \geq \bar{v} \geq v$, implying that $\hat{G}$ is also in G_{v_2} and $f_1(\hat{G}) \leq n - v_1$. Lemma 1 applied at iteration v of Algorithm 2 guarantees that $f_3(\bar{G}) \geq f_3(\hat{G})$. On the other hand $f_3(\bar{G}) \leq f_3(\hat{G})$, so that $f_3(\bar{G}) = f_3(\hat{G})$. The latter equality together with $f(\hat{G}) \neq f(\bar{G})$ means that $f_k(\hat{G}) < f_k(\bar{G})$, for some $k \in \{1, 2\}$. Consider $f_1(\hat{G}) < f_1(\bar{G})$ (the other case is similar). Then $f_1(\hat{G}) = n - \hat{v}_1 \leq n - (\bar{v}_1 + 1)$. This inequality together with $\bar{v}_2 \leq \hat{v}_2$ guarantees that the Steiner tree $\hat{G}$ can be found at iteration $v^+ = (\bar{v}_1 + 1, \bar{v}_2)$. Thus if iteration v^+ is executed ($R_{v^+} = 1$), the optimality of Algorithm 1 applied to this iteration results in $\tilde{G}$, so that $f_3(\tilde{G}) \geq f_3(\hat{G}) = f_3(\bar{G})$. We have $\tilde{v} \geq v^+$, where $\tilde{v}$ is the insertion position of $\tilde{G}$. Since $f_3(\tilde{G}) \geq f_3(\bar{G})$, $\tilde{v}_1 \geq \bar{v}_1 + 1$ and $\tilde{v}_2 \geq \bar{v}_2$, then, in *Phase 2* of Algorithm 2 when $i = \bar{v}_1 + 1$ and $j = \bar{v}_2$ we have $Q_{ij} \geq Q_{\tilde{v}} = f_3(\tilde{G}) \geq f_3(\bar{G}) = Q_{\bar{v}} = Q_{i-1j}$, which implies in $C_{\bar{v}} = \emptyset$, leading to a contradiction. Now suppose that iteration v^+ is not executed ($R_{v^+} = 0$). This means that at some iteration u such that $u \leq v^+$, a solution equivalent to $\tilde{G}$ is obtained, i.e., with an f_3 value and an insertion position greater than or equal to $f_3(\bar{G})$ and v^+, respectively. As above, $C_{\bar{v}} = \emptyset$. Therefore $\bar{G}$ is a Pareto-optimal solution. □

The next result shows that after Algorithm 2, the set of Steiner trees in C contains distinct objective vectors.

Lemma 2. *For any Steiner trees $\bar{G}$ and $\hat{G}$ in C obtained after Algorithm 2, we have $f(\bar{G}) \neq f(\hat{G})$.*

Proof. Consider $\bar{v}$ and $\hat{v}$, $\bar{v} \neq \hat{v}$, such that $C_{\bar{v}} = \bar{G}$ and $C_{\hat{v}} = \hat{G}$. By Algorithm 2, $f_1(\bar{G}) = n - \bar{v}_1$, $f_1(\hat{G}) = n - \hat{v}_1$, $f_2(\bar{G}) = d_{\bar{v}_2}$ and $f_2(\hat{G}) = d_{\hat{v}_2}$. Of course, $f_k(\bar{G}) = f_k(\hat{G}) \Leftrightarrow \bar{v}_k = \hat{v}_k$, $k = 1, 2$. But $\bar{v} \neq \hat{v}$ and, therefore, $f(\bar{G}) \neq f(\hat{G})$. □

Lemma 3. *During Algorithm 2 a Pareto-optimal solution is never deleted from matrix C.*

Proof. During *Phase 1* a solution $\hat{G}$ is only replaced by another solution $\bar{G}$ if $\hat{G}$ is dominated by $\bar{G}$. Now consider *Phase 2*. If it has executed $C_v \leftarrow \emptyset$, where $C_v = \hat{G} \neq \emptyset$ before this update, then we have $C_{\bar{v}} = \bar{G} \neq \emptyset$ with $f_3(\bar{G}) \geq f_3(\hat{G})$, for some $\bar{v} \geq v$ and $\bar{v} \neq v$. By Algorithm 2, $f_1(\bar{G}) = n - \bar{v}_1$, $f_1(\hat{G}) = n - v_1$, $f_2(\bar{G}) = d_{\bar{v}_2}$ and $f_2(\hat{G}) = d_{v_2}$. Since $\bar{v} \geq v$, with $\bar{v} \neq v$, then $n - \bar{v}_1 \leq n - v_1$ and $d_{\bar{v}_2} \leq d_{v_2}$ with at least one of the inequalities is strict, leading to $f_k(\bar{G}) \leq f_k(\hat{G})$, $k = 1, 2$, with $f_1(\bar{G}) < f_1(\hat{G})$ or $f_2(\bar{G}) < f_2(\hat{G})$. Then $f_k(\bar{G}) \leq f_k(\hat{G})$, $k = 1, 2$, $f_3(\bar{G}) \geq f_3(\hat{G})$ and $f(\bar{G}) \neq f(\hat{G})$, which means that $\hat{G}$ is dominated by $\bar{G}$. Therefore a Pareto-optimal is never deleted from C. □

The next theorem states that for each Pareto-optimal solution, the matrix C after Algorithm 2 contains one solution associated with the same objective vector.

Theorem 2. *Let G^* be any Pareto-optimal solution. After Algorithm 2 we have $C_{\bar{v}} = \bar{G}$ with $f(\bar{G}) = f(G^*)$, where $\bar{v}$ is such that $f_1(G^*) = n - \bar{v}_1$ and $f_2(G^*) = d_{\bar{v}_2}$.*

Proof. Graph G^* can be found at iteration $\bar{v}$. Then, if this iteration is executed ($R_{\bar{v}} = 1$), Lemma 1 ensures that the algorithm will find, at iteration $\bar{v}$, a solution $\bar{G}$ such that $f_3(\bar{G}) \geq f_3(G^*)$. Moreover $f_2(\bar{G}) \leq d_{\bar{v}_2} = f_2(G^*)$ and $f_1(\bar{G}) \leq n - \bar{v}_1 = f_1(G^*)$, because $\bar{G}$ was obtained in $G_{\bar{v}_2}$ with $h = n - \bar{v}_1$. Therefore $f_k(\bar{G}) \leq f_k(G^*)$, $k = 1, 2$ and $f_3(\bar{G}) \geq f_3(G^*)$, but since G^* is Pareto-optimal, then necessarily $f(\bar{G}) = f(G^*)$. This equality implies that $\bar{G}$ is also Pareto-optimal and $\bar{v}$ is exactly the $\bar{G}$'s insertion position. Therefore, by Lemma 3, after Algorithm 2 we have $C_{\bar{v}} = \bar{G}$. If iteration $\bar{v}$ is not executed ($R_{\bar{v}} = 0$), then we have found, at some previous iteration $v \leq \bar{v}$, a solution $\hat{G}$ with insertion position $\hat{v} \geq \bar{v}$. Since $v \leq \bar{v}$ then G^* can be find at iteration v, and thus, by Lemma 1, $f_3(\hat{G}) \geq f_3(G^*)$. The insertion position of $\hat{G}$ is $\hat{v} \geq \bar{v}$, then $f_1(\hat{G}) = n - \hat{v}_1 \leq n - \bar{v}_1 = f_1(G^*)$ and $f_2(\hat{G}) = d_{\hat{v}_2} \leq d_{\bar{v}_2} = f_2(G^*)$. Thus, as above, $f(\hat{G}) = f(G^*)$, implying that $\hat{G}$ is a Pareto-optimal solution and $\hat{v} = \bar{v}$, which means that $C_{\bar{v}} = \hat{G}$ after iteration v of *Phase 1*. But, by Lemma 3, $\hat{G}$ is never deleted from $C_{\bar{v}}$, so that at the end of Algorithm 2 we have $f(C_{\bar{v}}) = f(G^*)$. □

These results show that the set of Steiner trees given by C represents a minimal complete set of Pareto-optimal solutions. All solutions are Pareto-optimal (Theorem 1), each one is associated with a different objective vector (Lemma 2) and, for any Pareto-optimal, we have an equivalent solution in C, i.e., with the same objective vector (Theorem 2).

5 Computational experiments

The algorithm just described was implemented in C and ran on an Intel Core 2 Duo 2.0GHz with 3GB of RAM. In order to evaluate the performance of Algorithm 2, we present computational results on a set of randomly generated problems. These problems was generated as in Beasley [3], following the scheme outlined by Aneja [2]. They are defined by $n = |\mathcal{N}| = 2\,500,\ 7\,000,\ 13\,000$ and $20\,000$; $|\mathcal{T}| = n/100,\ n/10$ and $n/4$; and, average node degree equal to 4, 10 and 50. First a random spanning tree over $\mathcal{N}$ is generated, which guarantees that each instance corresponds to a connected graph. Then, additional random arcs are added to $\mathcal{A}$. For each $(i, j) \in \mathcal{A}$ we have $(j, i) \in \mathcal{A}$ with $d_{ij} = d_{ji}$. The set of terminal nodes $\mathcal{T}$ is randomly generated. For revenues r_i and lengths d_{ij}, integers were generated according to uniform distributions on the interval $[1, 100]$. The networks are represented using the forward star form (see Ahuja *et al.* [1]).

Table 1 provides the cardinality (#PO) of the minimal complete set of Pareto-optimal solutions, the number of iterations at which $R_v = 1$, i.e., how many times Algorithm 1 is called by Algorithm 2, and the CPU times in seconds. Values are averages over ten instances.

n	m	$\lvert\mathcal{T}\rvert$	#PO	Iterations	Seconds
2 500	10 000	25	75	80	0.043
2 500	10 000	250	4	7	0.007
2 500	10 000	625	2	5	0.006
2 500	25 000	25	72	123	0.085
2 500	25 000	250	15	60	0.046
2 500	25 000	625	8	49	0.039
2 500	125 000	25	36	131	0.164
2 500	125 000	250	14	103	0.128
2 500	125 000	625	7	94	0.115
7 000	28 000	70	18	20	0.042
7 000	28 000	700	2	4	0.020
7 000	28 000	1 750	1	4	0.018
7 000	70 000	70	52	89	0.212
7 000	70 000	700	17	50	0.131
7 000	70 000	1 750	8	37	0.103
7 000	350 000	70	39	132	0.547
7 000	350 000	700	16	105	0.444
7 000	350 000	1 750	8	93	0.386
13 000	52 000	130	37	40	0.149
13 000	52 000	1 300	4	6	0.042
13 000	52 000	3 250	1	3	0.034
13 000	130 000	130	72	102	0.506
13 000	130 000	1 300	18	41	0.227
13 000	130 000	3 250	8	28	0.167
13 000	650 000	130	45	135	1.138
13 000	650 000	1 300	18	102	0.883
13 000	650 000	3 250	11	99	0.836
20 000	80 000	200	13	16	0.119
20 000	80 000	2 000	1	3	0.058
20 000	80 000	5 000	1	3	0.056
20 000	200 000	200	59	73	0.670
20 000	200 000	2 000	18	31	0.322
20 000	200 000	5 000	3	9	0.125
20 000	1 000 000	200	43	121	1.874
20 000	1 000 000	2 000	18	97	1.565
20 000	1 000 000	5 000	10	85	1.367

Table 1: Summary of computational results.

The theoretical limit for the cardinality of the minimal complete set of Pareto-optimal solutions is $(n-1)m_1$. Since here $m_1 = 100$, then the upper bound for #PO is 249 900, 699 900, 1 299 900 and 1 999 900, for $n = 2\,500$, 7 000, 13 000 and 20 000, respectively. However, Table 1 shows that this value can be very small. It was always at least 3 332 times less than its theoretical upper bound. This is interesting for decision makers because the set of possible solutions is relatively small.

Another observation relates to the number of iterations executed by algorithm. As above, here we have the same theoretical limits, $(n-1)m_1$, and the number of iterations is very small. It is much closer to its lower bound #PO than to its upper bound $(n-1)m_1$. For $n = 13\,000$, for example, the number of iterations was never more than nine times #PO and always at least 9 628 times less than 1 299 900.

6 Conclusions

In this paper we have developed a polynomial algorithm for a tricriteria Steiner tree problem arising in the design of telecommunication networks. To our knowledge, this tricriteria problem has never been treated in the operational research literature. Our computational experiments show that large instances of this problem can be quickly solved by our algorithm. Instances with 20 000 nodes and 1 000 000 arcs were solved in less than two seconds. Additional constraints can be considered in this tricriteria problem. The problem is still easily solved if additional hop and bottleneck constraints are incorporated: $f_1 \leq \alpha$ and $f_2 \leq \beta$, respectively, where α, $\beta \in \mathbb{Z}_+$ with $\alpha < n-1$ and $\beta < d_1$. For this first constraint it is just necessary to change, in the initialization of Algorithm 2, $v_1 \leftarrow 1$ to $v_1 \leftarrow \bar{\alpha}$, so that $n - \bar{\alpha} = \alpha$. For the second constraint, arcs (i, j) with $d_{ij} > \beta$ should to be deleted from G before Algorithm 2 starts.

Acknowledgments This work was partly supported by Government of Canada, Graduate Students Exchange Program, by the National Council for Scientific and Technological Development of Brazil (CNPq), and by the Canadian Natural Sciences and Engineering Research Council under grant 39682-05. This support is gratefully acknowledged. Finally, thanks are due to the referees for their valuable comments.

References

[1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B. "Network Flows: Theory, Algorithms and Applications". Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] Aneja Y.P. "An integer linear programming approach to the Steiner problem in graphs". *Networks* 10, 167–178, 1980.

[3] Beasley J.E. "An SST-based algorithm for the Steiner problem in graphs". *Networks* 19, 1–16, 1989.

[4] Canuto, S.A., Resende, M.G.C., Ribeiro, C.C. "Local search with perturbations for the prize-collecting Steiner tree problem in graphs", *Networks* 38, 50–58, 2001.

[5] Chapovska, O., Punnen, A.P. "Variations of the prize-collecting Steiner tree problem". *Networks* 47, 199–205, 2006.

[6] Costa, A.M., Cordeau, J.-F., Laporte, G. "Steiner tree problems with profits". *INFOR* 44, 99–115, 2006.

[7] Costa, A.M., Cordeau, J.-F., Laporte, G. "Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints". *Networks* 53, 141–159, 2008.

[8] Cunha, A.S., Lucena, A., Maculan, N., Resende, M.G.C. "A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs". *Discrete Applied Mathematics* 157, 1198–1217, 2009.

[9] Gabow, H.N., Tarjan, R.E. "Algorithms for two bottleneck optimization problems". *Journal of Algorithms* 9, 411–417, 1988.

[10] Garey, M.R., Graham, R.L., Johnson, D.S. "The complexity of computing Steiner minimal trees". *SIAM Journal on Applied Mathematics* 32, 835–859, 1977.

[11] Gouveia, L. "Multicommodity flow models for spanning trees with hop constraints". *European Journal of Operational Research* 95, 178–190, 1996.

[12] Hakimi, S.L. "Steiner's problem in graphs and its implications". *Networks* 1, 113–133, 1971.

[13] Johnson, D.S., Minkoff, M., Phillips, S. "The prize collecting Steiner tree problem: theory and practice". *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, 2000.

[14] Lucena, A., Resende, M.G.C. "Strong lower bounds for the prize-collecting Steiner problem in graphs". *Discrete Applied Mathematics* 141, 277–294, 2004.

[15] Pinto, L.L., Pascoal, M. "Enhanced algorithms for tricriteria shortest path problems with two bottleneck objective functions". Submitted to *Computers & Operations Research*, 2009.

[16] Zelikovsky, A. "Bottleneck Steiner Tree Problems". In: *Encyclopedia of Optimization*, 2nd edition, C. A. Floudas and P. M. Pardalos (Eds.), Springer, New York, 311–313, 2009.

Anexo 4

Este anexo contém o artigo de Pinto, Bornstein e Maculan intitulado *A polynomial algorithm for the multi-objective bottleneck network problem with one MinSum objective function*, o qual foi submetido à *Networks* em dezembro de 2008.

A polynomial algorithm for the multi-objective bottleneck network problem with one MinSum objective function

Leizer de Lima Pinto
Cláudio Thomás Bornstein
Nelson Maculan*

COPPE/UFRJ – Federal University of Rio de Janeiro, Brazil
Dept. of Systems Engineering and Computer Science

Abstract: The paper presents multi-objective combinatorial optimization problems in graphs where at most one objective is a MinSum function and the other objectives are of the bottleneck type. A polynomial algorithm is developed which generates a minimal complete set of Pareto-optimal solutions. Optimality proofs are given.

Keywords: Multi-objective combinatorial optimization in graphs; Pareto-optimal solution; Bottleneck problem; Polynomial algorithm.

1) Introduction:

Combinatorial optimization problems in graphs generally consider two kinds of objective functions. The first can be called a *totalizing* function and involves the whole set of weights of arcs/nodes of a feasible solution. The best known example of this function is the MinSum function, where the objective is to minimize the sum of the weights of arcs of a path or a tree of a graph. Other possibilities are MaxSum, MinProd, MaxProd, etc.

The second kind of objective function, called a bottleneck function, considers just the maximum or the minimum weight of the solution. For example, a MinMax bottleneck function will try to find the path or tree whose arc with maximum weight is minimized. Another possibility is the MaxMin function. MaxMax and MinMin are of no significance for obvious reasons.

Here, a multi-objective problem will be examined where at most one objective is a totalizing function. All the other objectives are of the bottleneck type. Ehrgott and Gandibleux [9] present more than two hundred references in multi-objective combinatorial optimization. The reason for restricting the number of totalizing functions to one is that the generation of a minimal complete set of Pareto-optimal solutions for the shortest path, assignment and spanning tree multi-objective problems with more than one MinSum function is NP-complete (see Serafini [16] and Camerini *et al* [4]). Azevedo and Martins [1], Clímaco and Martins [5], Corley and Moon [6], Guerriero and Musmanno [11], Martins [13] and Tung and Chew [17] present non-polynomial algorithms for the multi-objective shortest path problem (MSPP) with more than one MinSum objective. Gandibleux *et al* [10] include a bottleneck function extending the algorithm presented by Martins [13].

Polynomial algorithms are presented by Hansen [12], Martins [14] and Berman *et al* [3] for the bicriterion MinMax-MinSum and by Pinto *et al* [15] for the tricriterion MaxMin-MinMax-MinSum shortest path problems. In both cases the goal is the generation of a minimal complete set of Pareto-optimal solutions.

Applications for the MSPP are given in Batta and Chiu [2], Current and Min [7] and Current and Marsh [8]. In Pinto *et al* [15] a tri-objective road transportation problem is presented, where the two bottleneck functions are associated with road quality and traffic density and cost minimization stands for the totalizing function.

*Corresponding author. *E-mail Address*: maculan@cos.ufrj.br. Federal University of Rio de Janeiro, COPPE/UFRJ, Dept. of Systems Engineering. P.O. Box 68511, 21941-972 Rio de Janeiro, RJ, Brazil.

Another possible application for the tri-objective shortest path problem is within communication networks. A sensor network may be represented by a graph and weights $p^1(i,j)$, $p^2(i,j)$ and $p^3(i,j)$ may be associated to each arc (i,j). $p^1(i,j)$ and $p^2(i,j)$ may represent the amount of available battery power and the average waiting time at node *i* respectively. $p^2(i,j)$ is a measure of overload of the system and represents the time between the in- and outflow of information from node *i*. $p^3(i,j)$ measures the transmission cost from *i* to *j*. A company may be interested in finding a route from a certain source *s* to a sink *t* so that the minimum battery power available at nodes of the route is maximized, maximum waiting time and total cost of the route are minimized. This corresponds to a MaxMin-MinMax-MinSum problem.

In this paper a polynomial algorithm will be presented for a graph optimization problem in which one objective is a MinSum function and the other *l* objectives are of the MinMax type. The algorithm here differs considerably from the *MMS* algorithm presented by Pinto *et al* [15]. First, trivially, $l+1$ objectives instead of just three are considered. Second, and more important, the algorithm presented in this paper is of a different type. In Pinto *et al* [15] spanning subgraphs are generated successively at each iteration in an order of increasing density. Here, on the contrary, spanning subgraphs are produced in an order of decreasing density. The advantage is that the non-availability of a path in a certain iteration of a certain loop makes it possible to skip the following iterations of this loop.

The third improvement presented in this paper concerns the set of restrictions. While in Pinto *et al* [15] a feasible solution is a path between two given nodes, here we handle a more general problem. Like Berman *et al* [3] given a graph $G=(N,M)$ it is just necessary that a feasible solution $s \subseteq M$ satisfies some property α. As long as a polynomial algorithm is available to optimize the single-objective problem with a totalizing function, the corresponding multi-objective problem will be solved in polynomial time.

Section two of the present paper presents some basic definitions, the notation and the *MMS-R* algorithm. Section three contains the optimality proofs, showing that the *MMS-R* algorithm does in fact generate a minimal complete set. Section four, the conclusions, gives a brief description of the computational performance of the *MMS-R* algorithm and some generalizations and extensions. As to our knowledge no exact polynomial algorithm is mentioned in the literature for the multi-objective graph optimization problem. Therefore, no comparison of computational efficiency is possible. However, a comparison of the performances of the *MMS* and *MMS-R* algorithms is made for the tricriterion shortest path problem.

2) Problem and algorithm:

Consider graph $G=(N,M)$ with $|N|=n$ and $|M|=m$. For each arc $(i,j)\in M$ let $p^k(i,j)\in \Re,\ k=1,2,\ldots,l+1$, be the *k*-th weight. Without any loss of generality suppose that the m_k different values of $p^k(.)$ in *M* are arranged in decreasing order, i.e. $p_1^k > p_2^k > \ldots > p_{m_k}^k$, for $k=1,2,\ldots,l$.

Let $v=(v_k)$ be an *l*-dimensional vector of indexes, where $1 \le v_k \le m_k$. Considering $M_v = \left\{(i,j)\in M \;\middle|\; p^k(i,j) \le p_{v_k}^k,\ k=1,2,\ldots,l\right\}$ leads to $G_v=(N,M_v)$ as a spanning subgraph of *G*. Remark that if *v* is so that $v_k=1,\ \forall k\in\{1,2,\ldots,l\}$ then $G_v=G$.

Let $s\subseteq M$ be a feasible solution satisfying some property α. For example, *s* may be a spanning tree or a path between two nodes of *G*. Let *S* be the set of all feasible solutions. The following multi-objective problem can be defined:

$$
\begin{aligned}
(P) \qquad & \text{minimize} \quad \max_{(i,j)\in s}\{p^k(i,j)\}, \quad k = 1,2,\ldots,l \\
& \text{minimize} \quad \sum_{(i,j)\in s} p^{l+1}(i,j) \\
& \text{subject to:} \quad s \in S.
\end{aligned}
$$

The first l objectives are of the MinMax type for the sake of simplicity. Any combination of MinMax or MaxMin will lead to the same kind of algorithm. Using MaxMin instead of MinMax leads to a reordering of weights.

Next, the definition of Pareto-optimal solution and minimal complete set is given. Let $q^s = (q_k^s) \in \Re^{l+1}$ be the *objective vector* associated with $s \in S$, where $q_k^s = \max_{(i,j)\in s}\{p^k(i,j)\}$ for $k = 1,2,\ldots,l$ and $q_{l+1}^s = \sum_{(i,j)\in s} p^{l+1}(i,j)$. We say that $s \in S$ *dominates* $\bar{s} \in S$ when $q^s \le q^{\bar{s}}$ with $q^s \ne q^{\bar{s}}$. A solution $s \in S$ is *Pareto-optimal* if there is no $\bar{s} \in S$ that dominates *s*. A set $S^* \subseteq S$ of Pareto-optimal solutions is a *minimal complete set* if for each two solutions $s', s'' \in S^*$ we have $q^{s'} \ne q^{s''}$ and for any Pareto-optimal solution $s \in S$ there exists $\bar{s} \in S^*$ so that $q^{\bar{s}} = q^s$.

The *MMS-R* algorithm generates a minimal complete set S^* of Pareto-optimal solutions for problem (P). In order to understand the algorithm some notation has to be defined:

- $C = (C_v)$ is a $m_1 \times m_2 \times \ldots \times m_l$ matrix which allows the determination of S^*. At the end of a run of the *MMS-R* algorithm, $C_v = s$ means that there exists a Pareto-optimal solution $s \in S^*$ corresponding to iteration *v*. Otherwise $C_v = \varnothing$.
- $Q = (Q_v)$ is also a $m_1 \times m_2 \times \ldots \times m_l$ matrix. If $C_v = s$, the matrix Q implicitly allows the determination of the vector q^s for iteration *v*. The value of q_{l+1}^s is given by Q_v. The other values of q_k^s for $k = 1,2,\ldots,l$ may be determined by the elements of vector *v*. It will be shown in theorem 1 that $q_k^s = p_{v_k}^k$. If $C_v = \varnothing$ then $Q_v = \infty$.
- Variables $ctrl_1, ctrl_2, \ldots, ctrl_l$ are not essential for understanding the basics of the algorithm. Replacing $ctrl_1, ctrl_2, \ldots, ctrl_l$ by their initial values $m_1, m_2, \ldots, m_l$ does not affect the optimality of *MMS-R*. Of course, in this case, the last set of instructions of the algorithm which update the value of $ctrl_k$ should be deleted. Variables $ctrl_1, ctrl_2, \ldots, ctrl_l$ merely allow the skipping of iterations where no feasible solution exists.
- *AlgS* is the algorithm which solves the single-objective problem with a totalizing function. In the case of (P), *AlgS* used in iteration *v* solves the corresponding MinSum problem for the spanning subgraph G_v using weights $p^{l+1}(.)$.
- e_j is the *l*-dimensional unit vector with the unity in the *j*-th position.

Next, the *MMS-R* algorithm is presented. The main structure of the algorithm is very easy to understand. It consists of l nested loops which enumerate successively all spanning subgraphs for weights $p^1(.), \ldots, p^l(.)$. Within a certain loop, graphs are generated in an order of decreasing density. At the core loop, *AlgS* tries to solve the single-objective MinSum problem for weight $p^{l+1}(.)$. In order

to generate S^* a test is made with the newly generated solution *s* detecting a possible improvement in one of the objective functions.

Do: $C_v := \varnothing$ *and* $Q_v := \infty$ *for all values of v, i.e. for* $v_k = 1,2,\ldots,m_k$, $k = 1,2,\ldots,l$, $ctrl_1 := m_1$ *and* $ctrl_2 := m_2$

For v_1 *from 1 to* $ctrl_1$ *do*

$ctrl_3 := m_3$

For v_2 *from 1 to* $ctrl_2$ *do*

$ctrl_4 := m_4$

For v_3 *from 1 to* $ctrl_3$ *do*

$\vdots$

$ctrl_l := m_l$

For v_{l-1} *from 1 to* $ctrl_{l-1}$ *do*

For v_l *from 1 to* $ctrl_l$ *do*

Use AlgS in G_v *considering function* $p^{l+1}(.)$

If no feasible solution exists then $ctrl_l := v_l - 1$

Else, let s be the solution obtained by AlgS

$C_v := s$, $Q_v := q_{l+1}^s$

For k from 1 to l do

If $v_k > 1$ *and* $Q_v = Q_{v-e_k}$ *then* $C_{v-e_k} := \varnothing$, $Q_{v-e_k} := \infty$

End For k

End Else

End For v_l

If $ctrl_l = 0$ *then* $ctrl_{l-1} := v_{l-1} - 1$

End For v_{l-1}

If $ctrl_{l-1} = 0$ *then* $ctrl_{l-2} := v_{l-2} - 1$

$\vdots$

End For v_3

If $ctrl_3 = 0$ *then* $ctrl_2 := v_2 - 1$

End For v_2

If $ctrl_2 = 0$ *then* $ctrl_1 := v_1 - 1$

End For v_1

Noticing that $m_1, m_2, \ldots, m_l \leq m$ it is easy to see that m^l is an upper limit of the number of iterations of the *MMS-R* algorithm. Each iteration implies in solving one single-objective problem with the help of *AlgS*. Supposing that the complexity of *AlgS* is $p(n)$ the complexity of *MMS-R* results in $O(m^l p(n))$.

3) Optimality proofs:

The following two theorems prove that after the end of a run of the *MMS-R* algorithm the set of solutions given by *C* represents a minimal complete set of Pareto-optimal solutions for problem (P). Theorem 1 shows that any solution in *C* is Pareto-optimal (part b) and that no two solutions in C have the same value of the objective vector (part a). Theorem 2 demonstrates that for any Pareto-optimal solution there exists a solution in *C* with the same value of the objective vector. These statements satisfy the definition of minimal complete set.

Theorem 1: *Let the values of matrix C be determined after the end of a run of the MMS-R algorithm. For any* $C_{\bar{v}} = s \neq \varnothing$ *the following two statements can be made:*

(a) $q_k^s = p_{\bar{v}_k}^k$, *for* $k = 1,2,\ldots,l$.

(b) s is a Pareto-optimal solution.

Proof:

(a) Since s was obtained in $G_{\bar{v}}$ we have $q_k^s \leq p_{\bar{v}_k}^k$, for $k = 1,2,\ldots,l$. The proof follows by contradiction. Suppose that $q_j^s < p_{\bar{v}_j}^j$ for some $j \in \{1,2,\ldots,l\}$. This means that $q_j^s \leq p_{\bar{v}_j+1}^j$. Therefore, s is a feasible solution in $G_{\bar{v}+e_j}$ what guarantees that *AlgS* will be able to find a solution $\bar{s}$ in iteration $\bar{v}+e_j$. The optimality of *AlgS* ensures that $q_{l+1}^{\bar{s}} \leq q_{l+1}^s$. On the other side, the fact that $M_{\bar{v}+e_j} \subseteq M_{\bar{v}}$ guarantees that $\bar{s}$ is in $G_{\bar{v}}$ and the optimality of *AlgS* in iteration $\bar{v}$ allows us to write that $q_{l+1}^s \leq q_{l+1}^{\bar{s}}$. The last two inequalities imply in $q_{l+1}^{\bar{s}} = q_{l+1}^s$. This equality together with the test made in the core loop (*For k* loop) for $k = j$ at iteration $\bar{v}+e_j$ results in $C_{\bar{v}} = \varnothing$, leading to a contradiction. Therefore, $q_k^s = p_{\bar{v}_k}^k$, $\forall k \in \{1,2,\ldots,l\}$.

(b) Again, the proof is by contradiction. Suppose that *s* is not a Pareto-optimal solution. Then there exists a feasible solution $\tilde{s}$ that dominates *s*, leading to $q^{\tilde{s}} \leq q^s$ with $q^{\tilde{s}} \neq q^s$. Using (a) allows us to write that $q_k^{\tilde{s}} \leq p_{\bar{v}_k}^k$, $\forall k \in \{1,2,\ldots,l\}$, what guarantees that $\tilde{s}$ is in $G_{\bar{v}}$. The optimality of *AlgS* in iteration $\bar{v}$ ensures that $q_{l+1}^s \leq q_{l+1}^{\tilde{s}}$. But the initial supposition was $q^{\tilde{s}} \leq q^s$. Consequently, $q_{l+1}^{\tilde{s}} = q_{l+1}^s$. This last equality together with $q^{\tilde{s}} \neq q^s$ and $q^{\tilde{s}} \leq q^s$ means that there exists $q_k^{\tilde{s}} < q_k^s$ for some $k \in \{1,2,\ldots,l\}$. Let *j* be such a value. From (a) we have $q_j^s = p_{\bar{v}_j}^j$ meaning that $q_j^{\tilde{s}} < p_{\bar{v}_j}^j$ and $q_j^{\tilde{s}} \leq p_{\bar{v}_j+1}^j$. Hence, $\tilde{s}$ is in $G_{\bar{v}+e_j}$ what guarantees that some solution $\bar{s}$ is obtained by *AlgS* in iteration $\bar{v}+e_j$. The optimality of *AlgS* in iteration $\bar{v}+e_j$ ensures that $q_{l+1}^{\bar{s}} \leq q_{l+1}^{\tilde{s}}$. But $q_{l+1}^{\tilde{s}} = q_{l+1}^s$, leading to $q_{l+1}^{\bar{s}} \leq q_{l+1}^s$. A similar reasoning to part (a) leads to the same contradiction, i.e. $C_{\bar{v}} = \varnothing$. Thus, *s* is a Pareto-optimal solution.

■

Theorem 2: *For any Pareto-optimal solution* s^*, *after the end of a run of the MMS-R algorithm, the matrix C holds a solution* s' *so that* $q^{s'} = q^{s^*}$.

Proof: Without any loss of generality let $\hat{v} = (\hat{v}_k)$ be so that $q_k^{s^*} = p_{\hat{v}_k}^k$ for $k = 1,2,\ldots,l$. Consequently, s^* is a feasible solution in $G_{\hat{v}}$ and it is possible to guarantee that in iteration $\hat{v}$ *AlgS* will obtain a solution s' so that $q_{l+1}^{s'} \leq q_{l+1}^{s^*}$. Since s' is obtained in $G_{\hat{v}}$ we have $q_k^{s'} \leq p_{\hat{v}_k}^k = q_k^{s^*}$ for

$k = 1,2,\ldots,l$. The *l*+1 last inequalities result in $q^{s'} \leq q^{s^*}$. Considering that s^* is a Pareto-optimal solution allows us to write that $q^{s'} = q^{s^*}$. Next, we prove by contradiction that s' belongs to *C*. Suppose that s' is excluded from *C* in a later iteration $\hat{v} + e_j$ for some $j \in \{1,2,\ldots,l\}$. This means that a solution $\bar{s}$ is obtained in iteration $\hat{v} + e_j$ so that $q^{\bar{s}}_{l+1} = q^{s'}_{l+1} = q^{s^*}_{l+1}$. Since $\bar{s}$ is a feasible solution in $G_{\hat{v}+e_j}$ we have $q^{\bar{s}}_k \leq p^k_{\hat{v}_k} = q^{s^*}_k$ for $k = 1,\ldots,j-1,j+1,\ldots,l$ and $q^{\bar{s}}_j \leq p^j_{\hat{v}_j+1} < p^j_{\hat{v}_j} = q^{s^*}_j$. Thus, $q^{\bar{s}} \leq q^{s^*}$ with $q^{\bar{s}}_j < q^{s^*}_j$, i.e. $\bar{s}$ dominates s^* contradicting the assumption that s^* is a Pareto-optimal solution. Hence s' is in *C*.

■

4) Conclusions:

As already mentioned, several modifications of problem (P) may be handled with slight changes of *MMS-R*. The MinSum objective function may be replaced by any other totalizing function or even by a bottleneck function. As long as *AlgS* is polynomial, the complexity of the resulting *MMS-R* algorithm will also be polynomial. Additionally, instead of taking MinMax functions for the first *l* objectives we may consider any combination of bottleneck functions (see Hansen [12] for the equivalence of MinMax and MaxMin).

Comparison of the computational performance of the *MMS* and *MMS-R* algorithms was made for the tricriterion shortest path problem. A heap implementation of Dijkstra was used for *AlgS*. Codes were written in *C* and both algorithms were run on a PC with a INTEL CORE 2 DUO processor at 1.73*GHz* with 1024*Mb* of RAM. Seventy graphs were randomly generated with *n* = 500, 1000 and 5000. For *n* = 500 and 1000 an outdegree of 10 was considered. For *n* = 5000 the graphs were produced with outdegrees of 5, 20 and 50. Weights $p^1(.)$ and $p^2(.)$ were randomly generated from the intervals [1, 10], [1, 50] and [1, 100], i.e. $m_1 = m_2 = 10, 50 \text{ and } 100$ respectively. Weights $p^3(.)$ were generated from the interval [1, 10000].

For the ten problems with *n* = 500 and $m_1 = m_2 = 10$, average times of *MMS* and *MMS-R* were 0.10 and 0.09 respectively. For the ten problems with *n* = 500 and $m_1 = m_2 = 100$, average times of *MMS* and *MMS-R* were 10.11 and 8.35 respectively. For the thirty problems with *n* = 5000 and $m_1 = m_2 = 10$, average times of *MMS* and *MMS-R* were 12.86 and 12.21 respectively and for ten problems with *n* = 5000 and $m_1 = m_2 = 50$ times were 136.17 and 111.59 respectively. All times are in seconds. As expected, computational times are strongly affected by the values of m_k (for further details see Pinto *et al* [15]). It seems reasonable to have $m_k \leq 100 \ \forall k$.

Although *MMS-R* results, in average, in almost half the number of iterations of *MMS*, times are nearly the same. The main reason is that iterations without a feasible solution spend almost no time of *MMS*. Another interesting result is that for a fixed number of nodes there is almost no increase of computational times with an increasing outdegree, i.e., times are not much affected by the density of the graph.

An important conclusion concerns the number of Pareto-optimal solutions, more specifically, the size of the minimal complete set S^*. If S^* is very large, decision makers will have problems in making the correct choices. Fortunately, for the seventy generated problems, S^* was kept within a manageable size. Maximum value of $|S^*|$ was 143 much below the theoretical upper limit $m_1 . m_2 = 10^4$ for $m_1 = m_2 = 100$. Of course this depends on the values of m_k. For bigger values, an

increase in S^* should be expected. This is a further argument for keeping the values of m_k low. For more details see Pinto *et al* [15].

With the present work the boundaries of multi-objective graph optimization problems seem to be defined quite precisely. With respect to the objectives, polynomiality is lost if more than one function is of the totalizing type. These problems are NP-complete and heuristics should be used for larger instances. For the other problems, i.e. if at least l out of l+1 functions are of the bottleneck type, *MMS-R* is able to find a minimal complete set in polynomial time.

A slight modification of *MMS-R* allows the generation of the *maximal complete set*, i.e. the set of all Pareto-optimal solutions (see Pinto *et al* [15]). Of course, for this case polynomiality is lost. However, our computational experience shows that there is not much difference between the minimal and the maximal complete set. For the 81 problems generated in [15] the greatest difference in size between the minimal and the maximal complete set was 2%. For 57 problems the size was exactly the same. Of course, these results depend on the data structure, particularly on the size of the interval from which $p^{l+1}(.)$ is generated.

With respect to the restrictions, deliberately feasibility was defined in a quite loose manner. Several multi-objective graph optimization problems may be handled by the *MMS-R* algorithm. Of course, polynomiality is lost if the single-objective problem with a totalizing function is not solvable in polynomial time.

References:

[1] J.A. Azevedo and E.Q.V. Martins, An algorithm for the multiobjective shortest path problem on acyclic networks, Investigação Operacional 11 (1991), 52-69.

[2] R. Batta and S.S. Chiu, Optimal obnoxious paths on a network: Transportation of hazardous materials, Operations Research 36 (1988), 84-92.

[3] O. Berman, D. Einav and G. Handler, The constrained bottleneck problem in networks, Operations Research 38 (1990), 178-181.

[4] P.M. Camerini, G. Galbiati and F. Maffioli, The complexity of multi-constrained spanning tree problems. Theory of Algorithms, Colloquium Pecs, L. Lovasz (Editor), North-Holland, Amsterdam (1984), 53-101.

[5] J.C.N. Clímaco and E.Q.V. Martins, On the determination of the nondominated paths in a multiobjective network problem, Methods in Operations Research 40 (1981), 255-258.

[6] H.W. Corley and I.D. Moon, Shortest paths in networks with vector weights, Journal of Optimization Theory and Applications 46 (1985), 79-86.

[7] J.R. Current and H. Min, Multiobjective design of transportation networks: Taxonomy and annotation, European Journal of Operational Research 26 (1986), 187-201.

[8] J.R. Current and M. Marsh, Multiobjective transportation network design and routing problems: Taxonomy and annotation, European Journal of Operational Research 103 (1993), 426-438.

[9] M. Ehrgott and X. Gandibleux, A survey and annotated bibliography of multiobjective combinatorial optimization, OR Spektrum 22 (2000), 425-460.

[10] X. Gandibleux, F. Beugnies and S. Randriamasy, Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function, A Quarterly Journal of Operations Research 4 (2006), 47-59.

[11] F. Guerriero and R. Musmanno, Label correcting methods to solve multicriteria shortest path problems, Journal of Optimization Theory and Applications 111 (2001), 589-613.

[12] P. Hansen, Bicriterion path problems. Multicriteria decision making: Theory and applications, Lecture Notes in Economics and Mathematical Systems 177, G. Fandel and T. Gal (Editors), Springer, Heidelberg (1980), 109-127.

[13] E.Q.V. Martins, On a multicriteria shortest path problem, European Journal of Operational Research 16 (1984), 236-245.

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

[15] L.L. Pinto, C.T. Bornstein and N. Maculan, The tricriterion shortest path problem with at least two bottleneck objective functions, European Journal of Operational Research, accepted 21 September 2008, (to appear).

[16] P. Serafini, Some considerations about computational complexity for multi objective combinatorial problems. Recent advances and historical development of vector optimization, Lecture Notes in Economics and Mathematical Systems 294, J. Jahn and W. Krabs (Editors), Springer, Berlin (1986), 222-232.

[17] C.T. Tung and K.L. Chew, A multicriteria Pareto-optimal path algorithm, European Journal of Operational Research 62 (1992), 203-209.