European Journal of Operational Research 242 (2015) 788–797

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An exact method for the biobjective shortest path problem for large-scale road networks

Daniel Duque, Leonardo Lozano, Andrés L. Medaglia[*]

Centro para la Optimización y Probabilidad Aplicada (COPA), Departamento de Ingeniería Industrial, Universidad de los Andes, Bogotá, Colombia

ARTICLE INFO

Article history:
Received 18 January 2013
Accepted 1 November 2014
Available online 6 November 2014

Keywords:
Routing
Multiobjective combinatorial optimization (MOCO)
Biobjective shortest path
Multiobjective shortest path
Pulse algorithm

ABSTRACT

The Biobjective Shortest Path Problem (BSP) is the problem of finding (one-to-one) paths from a start node to an end node, while simultaneously minimizing two (conflicting) objective functions. We present an exact recursive method based on implicit enumeration that aggressively prunes dominated solutions. Our approach compares favorably against a top-performer algorithm on two large testbeds from the literature and efficiently solves the BSP on large-scale networks with up to 1.2 million nodes and 2.8 million arcs. Additionally, we describe how the algorithm can be extended to handle more than two objectives and prove the concept on networks with up to 10 objectives.

1. Introduction

Consider a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N} = \{v_1, \ldots, v_i, \ldots, v_n\}$ is the set of nodes and $\mathcal{A} = \{(i, j) | v_i \in \mathcal{N}, v_j \in \mathcal{N}\}$ is the set of arcs. For all arcs $(i, j) \in \mathcal{A}$ let there be two nonnegative weights denoted by c_{ij} and t_{ij}. Henceforth, and without loss of generality, we refer to c_{ij} and t_{ij} as the cost and time of traversing arc $(i, j) \in \mathcal{A}$, respectively. The *Biobjective Shortest Path Problem* (BSP) is the problem of finding paths $\mathcal{P}$ from the start node $v_s \in \mathcal{N}$ to the end node $v_e \in \mathcal{N}$ that minimize two different (often conflicting) objective functions. The BSP can be formally defined as follows:

$$\min \mathbf{z}(\mathbf{x}) = (c(\mathbf{x}), t(\mathbf{x})) \tag{1}$$

s.t.,

$$\mathbf{x} \in \mathcal{X} \tag{2}$$

where $\mathbf{x}$ is a path $\mathcal{P}$ represented by a vector of (binary) arc flows x_{ij}, $(i, j) \in \mathcal{A}$; $c(\mathbf{x}) \triangleq \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij}$ is the cost of path $\mathbf{x}$; $t(\mathbf{x}) \triangleq \sum_{(i,j)\in\mathcal{A}} t_{ij}x_{ij}$ is the time of path $\mathbf{x}$; and $\mathcal{X}$ is the set of all paths from v_s to v_e. In (1) we (simultaneously) minimize the cost and time components of the vector function $\mathbf{z}(\mathbf{x})$. Since the existence of a path that simultaneously minimizes both objectives in (1) cannot be guaranteed, alternatively we seek for a set of paths with an acceptable tradeoff between the two objectives. Henceforth, we use functions $c(\cdot)$ and $t(\cdot)$ to represent the cost and time for complete solutions (i.e., a path $\mathcal{P}$ from v_s to v_e) or partial solutions (i.e., a path $\mathcal{P}$ from v_s to a certain node v_i), respectively.

This work aims to expand the body of knowledge of exact methods for the BSP. Our work shares its intuition with the *pulse algorithm* proposed by Lozano and Medaglia (2013) for the Constrained Shortest Path Problem (CSP), which has been successfully used as an algorithmic block for the multi-activity shift scheduling problem (Restrepo, Lozano, & Medaglia, 2012) and has been extended to the weight constrained shortest path problem with replenishment (Bolívar, Lozano, & Medaglia, 2014). To emphasize the fact that this work is an extension of a flexible solution framework, we purposely keep the *pulse* name in this paper.

The rest of the paper is organized as follows. Section 2 introduces relevant concepts for the BSP. Section 3 presents a literature review of the main solution strategies for the BSP. Section 4 introduces the pulse algorithm and the intuition behind it. Section 5 presents the core components of the algorithm. Section 6 compares the proposed algorithm against a top-performer algorithm by Raith (2010). Finally, Section 7 concludes the paper and outlines future work.

2. Basic concepts

This section introduces relevant concepts related to the biobjective shortest path problem. Let us recall that $\mathcal{X}$ is the set of all paths $\mathbf{x}$ from v_s to v_e. The image of any solution $\mathbf{x} \in \mathcal{X}$ on the objective space $\mathcal{Z}$ is a

[*] Corresponding author: Universidad de los Andes, Cr 1E No. 19A-10, ML711, Bogotá, Colombia Tel.: +57 13394949, ext:2880. URL http://wwwprof.uniandes.edu.co/~amedagli

E-mail address: amedagli@uniandes.edu.co, andres.medaglia@gmail.com (A. L. Medaglia).

http://dx.doi.org/10.1016/j.ejor.2014.11.003
0377-2217/© 2014 Elsevier B.V. All rights reserved.

vector denoted by $\mathbf{z}(\mathbf{x}) = (c(\mathbf{x}), t(\mathbf{x})) \in \mathcal{Z}$, where $c(\mathbf{x})$ and $t(\mathbf{x})$ are the values of each objective function (cost and time, respectively).

In the BSP, we look for a set of solutions that cannot improve one component of the objective vector $\mathbf{z}(\mathbf{x})$ without deteriorating the other one. These solutions are referred to as *efficient* solutions and are formally defined as follows:

Definition 2.1. A solution $\mathbf{x} \in \mathcal{X}$ is efficient if there does not exist another solution $\mathbf{x}' \in \mathcal{X}$ such that $c(\mathbf{x}') < c(\mathbf{x})$ and $t(\mathbf{x}') \leq t(\mathbf{x})$ or $c(\mathbf{x}') \leq c(\mathbf{x})$ and $t(\mathbf{x}') < t(\mathbf{x})$.

Efficient solutions could be either *supported* or *non-supported*. Supported solutions correspond to the optimal solutions of the mono-objective shortest path problem defined by the following linear (convex) combination of the objectives:

$$\min_{\mathbf{x}\in\mathcal{X}} \lambda_c c(\mathbf{x}) + \lambda_t t(\mathbf{x}) \tag{3}$$

where $(\lambda_c, \lambda_t) \in \Lambda$ are the weights given to the cost and time within the weight set $\Lambda = \{(\lambda_c, \lambda_t) \in \Re^2 | \lambda_c \geq 0, \lambda_t \geq 0, \lambda_c + \lambda_t = 1\}$. On the other hand, efficient solutions which are non-supported cannot be obtained by solving a shortest path problem with a weighted sum of the objectives as in (3).

Similar to the efficiency concept defined over the solution space, any given solution has a corresponding vector (point) in the objective space $\mathcal{Z}$ that can be either *dominated* or *non-dominated*. The following set of definitions clearly states the concepts of dominance and their relation with efficiency.

Definition 2.2. The image $\mathbf{z}(\mathbf{x})$ of an efficient solution $\mathbf{x}$ is said to be a non-dominated vector. If the solution is not efficient, then its image is a dominated vector in the objective space. The set of all non-dominated vectors is denoted by $\mathcal{Z}_N$.

Definition 2.3. Let $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ be two solutions representing feasible paths. If $c(\mathbf{x}') < c(\mathbf{x})$ and $t(\mathbf{x}') \leq t(\mathbf{x})$ or $c(\mathbf{x}') \leq c(\mathbf{x})$ and $t(\mathbf{x}') < t(\mathbf{x})$, then $\mathbf{z}(\mathbf{x})$ is said to be dominated by $\mathbf{z}(\mathbf{x}')$, and it is denoted by $\mathbf{z}(\mathbf{x}') \preceq \mathbf{z}(\mathbf{x})$.

Definition 2.4. Let $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ be two solutions representing feasible, but different paths $(\mathbf{x}' \neq \mathbf{x})$. If $c(\mathbf{x}') = c(\mathbf{x})$ and $t(\mathbf{x}') = t(\mathbf{x})$, then $\mathbf{x}'$ and $\mathbf{x}$ are said to be alternative paths.

Definition 2.5. The set of all efficient solutions X_E is called the true efficient set. Given an algorithm, the set of efficient solutions discovered so far at any given iteration is called the online efficient set and it is denoted by $\hat{X}_E$. If the algorithm is exact, once it meets its stopping criterion, $\hat{X}_E$ coincides with X_E. We also make the distinction between the online set of non-dominated vectors $\hat{\mathcal{Z}}_N$ and the true non-dominated set $\mathcal{Z}_N$.

3. Literature review

The BSP arises naturally on multiple real applications. In telecommunications, Clímaco and Pascoal (2012) presented network routing problems where it is necessary to find paths that minimize the total number of links while simultaneously minimize the bandwidth consumption. Pallotino and Scutellà (1998) stated that transportation problems often offer a tradeoff between two or more objectives, e.g., minimizing the arrival time to a final destination and the cost of the path. Müller-Hannemann and Weihe (2006) reported a railway routing problem that faces a compromise between monetary cost and travel time. Ehrgott, Wang, Raith, and Van Houtte (2012) presented a biobjective cyclist route choice model in which bike routes are designed based on the travel time and a suitability weight associated with each arc. Erkut and Verter (1998) presented a real-world hazmat transportation application, where the conflicting objectives are the path risk (i.e., probability of an accident) and its associated cost. Aside from its direct application, the BSP arises as a subproblem of combinatorial optimization models (Skriver & Andersen, 2000). In all the cases cited above, it is necessary to find a set of solutions that takes into account more than one objective simultaneously, rather than to find a single solution that focuses on a single objective.

Aside from its practical relevance, the BSP is also a challenging problem that is NP-hard (Serafini, 1986). Even though several researchers have proposed different methods for solving the BSP, there are two major solution strategies for the BSP: dynamic programming (DP) and ranking.

In the DP category, there are label correcting and label setting methods. Hansen (1980) and Martins (1984) were among the first authors who proposed a labeling approach for the BSP. The label correcting method is a straightforward extension of the mono-objective version (cf. Bertsekas, 1998), but with several labels at each node (Raith & Ehrgott, 2009). To find the efficient set of solutions, each node stores labels that represent tentative efficient paths. At the beginning, only the start node v_s is labeled. All labels at each node are extended through all the outgoing arcs, setting new labels over target nodes. When the label set of a node changes, the node is marked for reconsideration. When a node is reconsidered, all the dominated labels are deleted and the rest are extended. When the reconsideration heap is empty the algorithm finishes. Skriver and Andersen (2000) presented a label correcting algorithm that employs a node selection criterion for the reconsideration heap. Other versions, as the one presented by Guerriero and Musmanno (2001), employ a label selection criterion for the reconsideration heap. The label setting method works in a similar fashion. These algorithms always employ a label selection criterion and the main difference with label correcting is that only the selected label is extended through all the arcs. Raith and Ehrgott (2009) implemented a label setting algorithm using a binary heap for the labels storage. In this algorithm, the smallest lexicographically ordered label is selected from all nodes to be extended at each iteration. The extended label is compared with the labels at the target node of each arc and dominated labels are deleted. Several speedup strategies for DP approaches have been developed recently. Raith (2010) proposed bounded label correcting and setting algorithms. These bounded versions use the labels at the end node for the dominance test at each node (in addition to the node's own labels). Iori, Martello, and Pretolani (2010) presented a label setting policy that treats labels according to an aggregate function calculated for each label. Demeyer, Goedgebeur, Audenaert, Pickavet, and Demeester (2013) used the same idea of a bounded labeling algorithm (Raith, 2010) in a unidirectional/bidirectional label setting algorithm. The bidirectional DP extends labels forward from the start node and backward from the end node. When forward and backward labels reach the same node, the labels are combined and added into the online non-dominated set. Both searches are aborted as soon as the there are no forward nor backward labels dominating solutions of the online non-dominated set (i.e., there are no promising labels to extend). Even though Demeyer et al. (2013) reported speedups of the bidirectional DP over the unidirectional version on the common testbed instances used by Raith (2010) and Demeyer et al. (2013), computational times are better for the labeling approach of Raith (2010). Müller-Hannemann and Schnee (2007) and Disser, Müller-Hannemann, and Schnee (2008) also proposed speedup techniques for DP algorithms that exploit particular characteristics of time-dependent networks used in railroad routing.

In the ranking category, the near shortest path (NSP) method finds all the paths within a certain deviation from the shortest path length found by solving the weighted sum problem associated with the BSP. Carlyle and Wood (2005) presented a method that, besides its remarkable performance for solving the near shortest path problem, it outperforms other specialized algorithms solving the k-shortest path problem. Raith and Ehrgott (2009) compared different solution strategies including label setting and label correcting approaches, the near shortest path method, and a two-phase method based on the