Parametric search for the bi-attribute concave shortest path problem

Yuli Zhang
Department of Industrial Engineering, Tsinghua University, Beijing 100084, P.R. China, zhangyuli@mail.tsinghua.edu.cn

Zuo-Jun Max Shen
Department of Industrial Engineering & Operations Research, and
Department of Civil and Environmental Engineering, University of California, Berkeley, CA 94720, USA,
maxshen@berkeley.edu

Shiji Song
Department of Automation, TNList, Tsinghua University, Beijing 100084, P.R. China, shijis@mail.tsinghua.edu.cn

A bi-attribute concave shortest path (BC-SP) problem seeks to find an optimal path in a bi-attribute network that minimizes a linear combination of two path costs, one of which is evaluated by a nondecreasing concave function. Due to the nonadditivity of its objective function, Bellman's principle of optimality does not hold. This paper proposes a parametric search method to solve the BC-SP problem, which only needs to solve a series of shortest path problems, i.e., the parameterized subproblems (PSPs). Several techniques are developed to reduce both the number of PSPs and the computation time for these PSPs. Specifically, we first identify two properties of the BC-SP problem to guide the parametric search using the gradient and concavity of its objective function. Based on the properties, a monotonic descent search (MDS) and an intersection point search (IPS) are proposed. Second, we design a speedup label correcting (LC) algorithm, which uses optimal solutions of previously solved PSPs to reduce the number of labeling operations for subsequent PSPs. The MDS, IPS and speedup LC techniques are embedded into a branch-and-bound based interval search to guarantee optimality. The performance of the proposed method is tested on the mean-standard deviation shortest path problem and the route choice problem with a quadratic disutility function. Experiments on both real transportation networks and grid networks show that the proposed method reduces the computation time of existing algorithms by one to two orders of magnitude.

Key words: bi-attribute concave shortest path; parametric search; interval search; speedup label correcting

1. Introduction

1.1. Motivation

The shortest path problem plays a fundamental role in the fields of transportation science, telecommunications, logistics, etc. The classical shortest path problem aims at finding an optimal path with the minimum travel time. However, in practice, usually more than one attributes of paths influence the optimal routing decisions. Furthermore, a risk-averse decision maker in a stochastic network considers not only the expected travel time but also the path reliability, such as the travel time variability. To strike a balance between two different criteria, optimal bicriterion routing decisions need to be made. To this end, this paper studies a bi-attribute concave shortest path (BC-SP) model of the form: $\min\{h(\nu(p)) + \mu(p) : p \in P\}$, where $h : R \to R$ is a non-decreasing differentiable concave function, P is the set of loop-less paths from an origin to a destination and $(\nu(p), \mu(p))$ are nonnegative attributes of a path $p \in P$.

An important application of the BC-SP model is the mean-standard deviation shortest path (MSD-SP) problem (Wu and Nie 2011, Khani and Boyles 2015), where h is the square root function, ν_{ij} and μ_{ij} represent the variance and mean of the random travel time of a path p. The MSD-SP problem seeks to find a reliable path, which minimizes the sum of the mean and weighted standard deviation of path travel time (Noland and Polak 2002, Zeng et al. 2015, Shahabi and Boyles 2015, Zhang et al. 2016a).

Besides the MSD-SP problem, the BC-SP model also arises in multi-attribute transportation networks. For example, as pointed out by Mirchandani and Wiecek (1993), the transportation cost is usually a non-decreasing concave function of travel distance, and thus considering both travel time and transportation cost gives rise to a BC-SP problem. Other applications of the BC-SP model include the traffic equilibrium problem (Gabriel and Bernstein 1997, Zhang et al. 2011), the multi-agent network problem (Gabriel and Bernstein 2000) and the nonlinear congestion pricing problem (Lawphongpanich and Yin 2012).

The BC-SP problem belongs to discrete concave minimization problems, which are usually hard to solve exactly (Tuy 1998). Due to its nonadditivity, Bellman's principle of optimality does not hold for the BC-SP problem, i.e., subpaths of optimal paths are not necessarily optimal. In the remainder of this section, we first review existing solution methods, and then outline the proposed method and our contributions.

1.2. Literature review

1.2.1. Enumeration Methods The monotonicity of the objective function indicates that optimal paths of the BC-SP problem are non-dominated (Pareto) paths. A direct method to solve the BC-SP problem is to generate all non-dominated paths and select the optimal one. Enumerating all non-dominated paths has been extensively studied as the bicriterion short path (BSP) problem

since the seminal work of Hansen (1980). Solution methods include the label setting (LS), label correcting (LC), ranking and two phase methods. LS and LC methods generalize Dijkstra's and Bellman-Ford-Moore's shortest path algorithms to the multi-criteria setting, respectively. Most recent studies focus on speeding up labeling methods using dominance relations, such as Iori et al. (2010) (LS), Raith (2010) (LS, LC) and Demeyer et al. (2013) (LS). In addition, Lozano and Medaglia (2013) and Duque et al. (2015) propose a depth-first labeling method, i.e., the pulse algorithm. Ranking method utilizes k-shortest path (Climaco and Martins 1982) or near shortest path algorithms (Raith and Ehrgott 2009). Two phase methods first generate the extreme non-dominated paths using a parametric analysis and then generate the remainders by LS, LC or k-shortest paths algorithms (Mote et al. 1991). There are numerous studies on the BSP problem. A comprehensive overview is beyond the scope of this paper. The reader is referred to recent review papers, including Raith and Ehrgott (2009), Climaco and Pascoal (2012) and Paixao and Santos (2013).

The concavity of the objective function further shows that optimal paths of the BC-SP problem belong to the extreme non-dominated path set, which is a subset of the non-dominated path set. Therefore, it is sufficient to enumerate the extreme non-dominated paths. Henig (1986) proposes three methods to generate all extreme non-dominated paths: the sequential parametric analysis (SPA), the bidirectional parametric analysis (BPA) and dynamic programming (DP). The SPA method is a customized parametric linear programming method, which iteratively increases a parameter. The BPA method updates the parameter from both left and right directions. DP uses a generalized Bellman's principle of optimality. Recently, Sedeno-Noda and Raith (2015) improve the SPA method and propose a Dijkstra-like method computing all extreme non-dominated paths. To solve the MSD-SP problem, Khani and Boyles (2015) propose two iterative labeling (IL) algorithms: the basic one is similar to the SPA method and the improved one reduces the computation time by partially updating labels.

1.2.2. Implicit Enumeration Methods Although enumeration methods can solve the BC-SP problem, Hansen (1980) shows that the number of non-dominated paths may grow exponentially in n, and Carstensen (1983) further shows that the number of extreme non-dominated paths could be $2^{\Omega(\log^2 n)}$, where n is the number of nodes. One approach to reduce the number of enumerations is to exploit dominance properties. Hutson and Shier (2009) propose a modified labeling algorithm based on an e-dominance condition for a general bicriterion convex-concave shortest path problem. Chen et al. (2013) propose multi-criteria LS and A* algorithms based on a fast dominance check for the MSD-SP problem.

Another implicit enumeration method is the parametric search method, which has been widely used to solve discrete optimization problems (Tuy 1998). The parametric search method is similar

to the BPA method, but it only needs to enumerate partial extreme non-dominated paths by solving a series of parameterized subproblems (PSPs): (P_λ) $\min\{\lambda\nu(p) + \mu(p) : p \in P\}$ where $\lambda \in [0, +\infty)$. Henig (1986) suggests a branch-and-bound (B&B) based parametric search to find the optimal non-dominated extreme path. To find the best non-dominated path through an interaction with decision makers, Current et al. (1990) and Coutinho-Rodrigues et al. (1999) combine the parametric search with constrained shortest path and k-shortest paths approaches. Xie and Waller (2012) propose a polynomial time parametric algorithm to approximate non-dominated paths of the BSP problem. Nikolova (2009) uses the parametric search to solve the BC-SP problem, which generates new parameters using the perpendicular method (Dial 1979, Henig 1986). Zhang et al. (2016b) further propose a B&B based interval search algorithm, which utilizes lower bounds on the objective function over parameter intervals to reduce the number of PSPs. To reduce the number of PSPs, both Nikolova (2009) and Khani and Boyles (2015) design heuristics to guide the parametric search. However, optimality of these heuristics can not be guaranteed.

The deficiencies of existing parametric search methods for the BC-SP problem are twofold. First, they only use a lower bound to speed up the parametric search, and may need to solve a possibly large number of PSPs. An approach to address this issue is to use the gradient of the objective function to guide the parametric search. For example, gradient based line search methods have been proposed to find optimal extreme non-dominated paths of the bi-attribute convex shortest path problem, where h is a non-decreasing convex function (Henig 1986, Mirchandani and Wiecek 1993, Murthy and Sarkar 1996, Tsaggouris and Zaroliagis 2004). The effectiveness of these methods depends on the fact that the optimal value of (P_λ) is a unimodal function in λ for the bi-attribute convex shortest path problem (Henig 1986). However, Mirchandani and Wiecek (1993) show that the unimodality does not hold for the BC-SP problem. Second, existing parametric search methods solve each PSP from scratch, and the previously obtained shortest paths have not been used to speed up the solving process of the subsequent PSPs. The focus of this paper is to present an efficient parametric search method to overcome these deficiencies.

1.2.3. Other Methods To solve the MSD-SP problem, Shahabi et al. (2013) and Shahabi and Boyles (2015) propose an outer approximation (OA) method, and Xing and Zhou (2011) propose a Lagrangian substitution based method. Both methods can handle link travel time correlation. However, in each iteration, the OA method has to solve a constrained shortest path problem, which is NP-complete (Garey and Johnson 1979). There exists a duality gap for the second method due to the non-convex feasible region and objective function.

Approximate solution methods for concave minimization problems have been proposed by Nikolova (2010). For general bicriterion shortest path problems, where h is a general nonlinear continuous function, Chen and Nie (2013) approximate the nonlinear function by a piecewise linear counterpart, and then solve each linear subproblem sequentially.

1.3. Proposed method

This paper adopts the parametric search method to solve the BC-SP problem. To improve the efficiency of the parametric search method, we further develop several techniques to guide the parametric search and design a speedup LC algorithm for the PSPs.

This paper enhances existing parametric search methods by identifying two properties of the BC-SP problem and proposing efficient parametric search procedures. To obtain these properties, we reformulate the BC-SP problem as a two-dimensional concave minimization problem in the ν-μ space and show that there exists an "optimal" parameter such that optimal solutions for the corresponding PSP are also optimal for the BC-SP problem. Our first property, named monotonic descent property, indicates that gradient of the objective function can be iteratively used to find better paths and narrow the range of the optimal parameter. Although gradient has been used to find locally optimal solutions of concave minimization problems (Tuy 1998), the monotonic descent property has not been exploited to speed up the parametric search for the BC-SP problem (Nikolova et al. 2006, Nikolova 2009, Khani and Boyles 2015). A monotonic descent search (MDS) procedure is further proposed based on this property.

To avoid being trapped in locally optimal paths, we use a bi-directional parametric search strategy and further derive an intersection point property, which shows how the information collected in one direction can be used to speed up the parametric search in the other direction. Based on these two properties, an intersection point search (IPS) procedure is proposed. The IPS procedure is unlikely to be trapped in locally optimal paths, since it terminates only when the search in both directions finds locally optimal paths with the same objective function value. Our experiments also show that the IPS procedure solves all the test instances of the MSD-SP problem and the route choice problem with a quadratic disutility function (RC-QDF) (Chen and Nie 2013, Wu and Nie 2011). Finally, we further embed both the MDS and IPS procedures into a B&B based interval search algorithm to guarantee global optimality in theory.

Another contribution of this paper is a speedup LC algorithm for the PSPs. Specifically, we show how to use shortest paths of previously solved PSPs to estimate both upper and lower bounds on the shortest path length of the subsequent PSPs. The speedup LC algorithm is designed to reduce the number of labeling operations, and is further embedded into the interval search algorithm. We show the proposed speedup LC algorithm runs in $\mathcal{O}(nm)$ time in the worst case, and experimental results show that it significantly reduces the computation time for the PSPs.

The major contributions of this paper can be summarized as follows:

1. Two properties of the BC-SP problem are identified. MDS and IPS procedures are proposed to improve the efficiency of existing parametric search methods.

2. A speedup LC algorithm is proposed to solve the PSPs efficiently.

3. An interval search algorithm is proposed to guarantee global optimality.

4. Experiments for the MSD-SP and RC-QUF problems show that the proposed method reduces the computation time of the improved IL algorithm (Khani and Boyles 2015) and OA algorithm (Shahabi et al. 2013, Shahabi and Boyles 2015) by one to two orders of magnitude.

The remainder of this paper is structured as follows. The next section provides formal formulations of the BC-SP problem. Section 3 presents the proposed parametric search method. Section 4 gives the speedup LC algorithm. Section 5 reports and discusses experimental results. Concluding remarks and future work are provided in Section 6.

2. Problem Statement

In this section, we first provide a list of notation used in this paper, and then describe the BC-SP problem and the basic idea of the parametric search method.

2.1. Notation

Network	
N	set of nodes, where $\|N\| = n$,
A	set of arcs, where $\|A\| = m$,
$\bar{A}$	set of the reversed arcs of A, i.e., $\bar{A} = \{ji : i \in N, j \in N, ij \in A\}$,
s, t	the origin and destination nodes in N,
ν_{ij}, μ_{ij}	two nonnegative attributes of arc $ij \in A$,
$G(N, A)$	a directed and connected network composed by N and A,
$G^\lambda(N, A)$	a parameterized network of $G(N, A)$ with the arc cost $\lambda\nu_{ij} + \mu_{ij}$, where $\lambda \geq 0$,
$G^\lambda(N, \bar{A})$	the reversed network of $G^\lambda(N, A)$,
Problems	
(P)	the considered bi-attribute concave shortest path problem in $G(N, A)$,
(P_λ)	a shortest path problem from s to t in $G^\lambda(N, A)$,
f^*	the optimal value of the considered problem (P),
Paths	
P	set of loop-less paths from s to t,
p	a loop-less path from s to t, i.e., $p \in P$
P^*	set of optimal paths (solutions) of the considered problem (P),
P^λ	set of shortest paths from s to t in $G^\lambda(N, A)$,
p^k_{it}	a shortest path from i to t in $G^{\lambda_k}(N, A)$, where $\lambda_k \geq 0$,
Labels	
$(\nu(p), \mu(p))$	the label of a path p,
H	set of labels of paths in P,
H_n	set of non-dominated labels,
H_e	the set of extreme non-dominated labels,
H^*	set of labels of paths in P^*,
H^λ	set of labels of paths in P^λ,
(x_λ, y_λ)	a label in H^λ, which corresponds to at least a path in P^λ,
d^λ_{it}	the shortest path length from i to t in $G^\lambda(N, A)$.

2.2. Problem formulation

Consider a directed and connected network $G(N, A)$ consisting of a set of nodes N ($|N| = n$) and a set of arcs A ($|A| = m$). Suppose that P represents the set of loop-less paths from a specified origin node $s \in N$ to a specified destination node $t \in N$. Let (ν_{ij}, μ_{ij}) be a pair of nonnegative real-valued attributes associated with the arc $ij \in A$. A two-dimensional label $(\nu(p), \mu(p))$ associated with a path $p \in P$ is given by $\nu(p) = \sum_{ij \in p} \nu_{ij}$ and $\mu(p) = \sum_{ij \in p} \mu_{ij}$. In the following, we simply refer to a loop-less path as a path.

We are interested in finding a path $p \in P$ to minimize a bi-attribute concave function, that is,

$$\text{(P)} \quad \min \left\{ h\big(\nu(p)\big) + \mu(p) : p \in P \right\},$$

where $h : R \to R$ is a non-decreasing differentiable concave function. An example of the BC-SP model is the risk-averse MSD-SP problem (Wu and Nie 2011, Khani and Boyles 2015), where $h = \theta\sqrt{x}$ ($\theta \geq 0$), ν_{ij} and μ_{ij} represent the variance and mean of the random travel time on each arc $ij \in A$.

We first introduce related notation and definitions. Let $H = \{(x, y) : x = \nu(p), y = \mu(p), p \in P\} \subseteq R^2$ be the set of labels of paths in P and $\text{conv}(H)$ be the convex hull of H.

Definition 1. A path $p \in P$ dominates another path $p' \in P$ if $\nu(p) \leq \nu(p')$, $\mu(p) \leq \mu(p')$ and $(\nu(p), \mu(p)) \neq (\nu(p'), \mu(p'))$.

Definition 2. A path $p \in P$ and its associated label $(\nu(p), \mu(p))$ are said to be non-dominated if there exists no path $p' \in P$ that dominates p.

Definition 3. A non-dominated path $p \in P$ and its associated label $(\nu(p), \mu(p))$ are said to be extreme non-dominated if $(\nu(p), \mu(p))$ is an extreme point of $\text{conv}(H)$, i.e., $(\nu(p), \mu(p))$ does not lie in any open line segment joining two points of $\text{conv}(H)$.

Let $H_n, H_e \subseteq H$ denote sets of all non-dominated and extreme non-dominated labels, respectively. It is easy to see that $H_e \subseteq H_n$ and the extreme non-dominated labels lie on the lower-left boundary of $\text{conv}(H)$. Figure 1 gives an illustration of H_n, H_e and H.

As pointed out by Chen et al. (2013), Bellman's principle of optimality does not hold for the BC-SP problem due to the nonadditivity of its objective function. Since the objective function of (P) is non-decreasing in both ν and μ, to solve the BC-SP problem, it is sufficient to enumerate all non-dominated paths by solving the so-called bicriterion shortest path problem. However, the number of such paths may be huge. Instead of enumerating these non-dominated paths, we solve (P) by a parametric search in the ν-μ space.

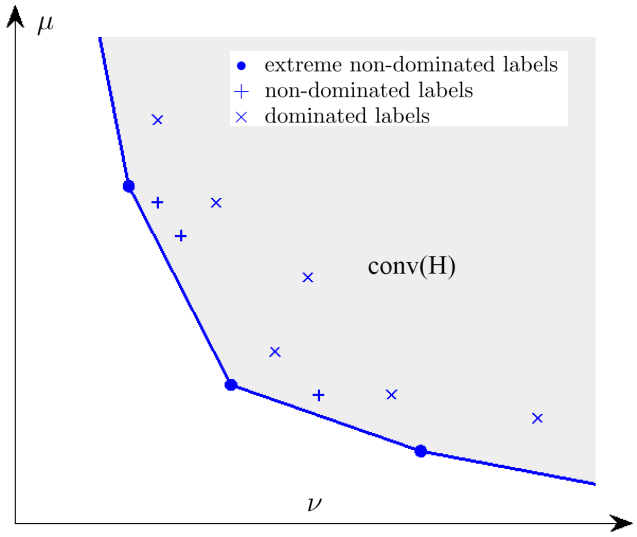

Figure 1 **Illustration of label sets.**

2.3. Basic idea of parametric search

Let $f(x,y)=h(x)+y$. Since f is a concave function over R^2, we have

$$\min_{p\in P}\{h(\nu(p))+\mu(p)\}=\min_{(x,y)\in H} f(x,y)=\min_{(x,y)\in \mathrm{conv}(H)} f(x,y),$$

where the last equality uses the fact that the concave function f over the compact convex set $\mathrm{conv}(H)$ attains its maximum at an extreme point of $\mathrm{conv}(H)$ (Horst et al. 2000), and all extreme points of $\mathrm{conv}(H)$ belong to H.

Therefore, to solve (P), it is sufficient to find an extreme point of $\mathrm{conv}(H)$, i.e., an extreme non-dominated label in H_e, which solves the following two-dimensional concave minimization problem:

$$\text{(P1)}\quad \min\{f(x,y):(x,y)\in \mathrm{conv}(H)\},$$

and then any path with this label is an optimal solution of (P).

To solve (P1), this paper considers the following PSPs:

$$(\mathrm{P}_\lambda)\quad \min_{(x,y)\in \mathrm{conv}(H)}\{\lambda x+y\}=\min_{(x,y)\in H}\{\lambda x+y\}=\min_{p\in P}\{\lambda\nu(p)+\mu(p)\},\quad \lambda\ge 0$$

where the second equality uses the fact that a linear programming problem attains its minimum at an extreme point if its minimum exists. Although solving (P_λ) does not necessarily provide an extreme non-dominated label, we will see that there exists an "optimal" parameter λ^* such that optimal solutions for (P_{λ^*}) are also optimal for (P); see Proposition 1.

To design the parametric search method, we use the following two observations:

(1) Any extreme non-dominated label is an optimal solution of (P_λ) for some $\lambda\ge 0$.

(2) For a fixed value of λ, (P_λ) can be solved by the classical labeling methods as a shortest path problem with a weighted arc cost $\lambda\nu_{ij} + \mu_{ij}$ on $ij \in A$. Therefore, solving (P_λ) not only provides an extreme non-dominated label, but also the corresponding extreme non-dominated path.

Based on these two observations, Khani and Boyles (2015) design IL algorithms for the MSD-SP problem, which enumerate all extreme non-dominated paths by varying λ from 0 to ∞. However, since the number of extreme non-dominated paths could be super-polynomial (Carstensen 1983), the IL algorithms may be inefficient. To avoid enumerating all the extreme non-dominated paths, this paper exploits both the gradient and concavity of f to design more efficient parametric search procedures, and proposes a speedup LC algorithm for (P_λ).

In the following, let P^* and H^* be the set of optimal paths and labels of (P), respectively. Let P^λ and H^λ be the sets of optimal paths and labels of (P_λ), respectively. Let f^* be the optimal value of (P).

3. Parametric search method

Khani and Boyles (2015) show that $P^* \subseteq \cup_{\lambda\geq 0} P^\lambda$ and solve the MSD-SP problem by enumerating paths in $\cup_{\lambda\geq 0} P^\lambda$. An important question is to find an "optimal" λ^* such that $P^{\lambda^*} \subseteq P^*$, and thus any path in P^{λ^*} is an optimal solution of (P). The following proposition provides a sufficient condition for such λ^*. A more general version of the following proposition can be found in Sniedovich (1986).

PROPOSITION 1. *For any optimal path $p^* \in P^*$ with the label (ν^*, μ^*), let $\lambda^* = h'(\nu^*)$, where $h'(x)$ is the derivative of h with respect to x, then $P^{\lambda^*} \subseteq P^*$ and $H^{\lambda^*} \subseteq H^*$.*

Proof: For any $\bar{p} \in P^{\lambda^*}$ with the label $(\bar{\nu}, \bar{\mu}) \in H^{\lambda^*}$, from the definition of P^{λ^*}, we have $\lambda^*\nu^* + \mu^* \geq \lambda^*\bar{\nu} + \bar{\mu}$, that is, $\mu^* - \bar{\mu} \geq \lambda^*(\bar{\nu} - \nu^*)$. From the concavity of h, we have that $\lambda^*(\bar{\nu} - \nu^*) = h'(\nu^*)(\bar{\nu} - \nu^*) \geq h(\bar{\nu}) - h(\nu^*)$. Therefore, we have $h(\bar{\mu}) + \bar{\nu} \leq h(\nu^*) + \mu^* = f^*$, that is, $\bar{p} \in P^*$ and $(\bar{\nu}, \bar{\mu}) \in H^*$. □

Proposition 1 shows that the "optimal" λ^* can be selected as the gradient of f at any $(\nu^*, \mu^*) \in H^*$. Although initially H^* is unknown, an estimation of ν^* can narrow the range of λ^*. In general, a lower bound on ν^* can be given as $\nu_L := \min_{p\in P} \nu(p)$. An upper bound on ν^* can be given as $\nu_U := \nu(p')$, where p' is an optimal path in P^0. For networks with special structures, bounds on ν^* may be easily estimated from the value of ν_{ij}. Due to the concavity of h, $h'(\nu)$ is non-increasing in ν. Therefore, the "optimal" λ^* lies in the interval $[\lambda_L, \lambda_U]$, where $\lambda_L := h'(\nu_U)$ and $\lambda_U := h'(\nu_L)$.

Proposition 1 also shows that any optimal path p^* with the label (ν^*, μ^*) has the property: $p^* \in P^{h'(\nu^*)}$. We refer to paths with this property as locally optimal paths, i.e., a path $p \in P$ and its associated label (ν, μ) are said to be locally optimal if $p \in P^{h'(\nu)}$. In Subsection 3.1 and 3.2, parametric search techniques are proposed to find locally optimal paths based on the gradient and concavity of f. In Subsection 3.3, an interval search algorithm is designed to find a globally optimal path.

3.1. Monotonic descent search

This subsection gives a monotonic descent search to find locally optimal paths. The following result is widely used in bicriterion optimization problems (Xie and Waller 2012) and shows that as the value of λ increases, the corresponding extreme non-dominated label moves from the lower right to upper left along the lower-left boundary of conv(H).

LEMMA 1. *If $\lambda_2 > \lambda_1 \geq 0$ and $(x_{\lambda_i}, y_{\lambda_i}) \in H^{\lambda_i}$ for $i = 1, 2$, we have $x_{\lambda_1} \geq x_{\lambda_2}$ and $y_{\lambda_1} \leq y_{\lambda_2}$.*

Proposition 2 shows how to find a locally optimal label from any $\lambda_1 \geq 0$ by utilizing the gradient of f. A locally optimal path can also be obtained by solving the corresponding shortest path problem.

PROPOSITION 2. *For any $\lambda_1 \geq 0$ and $(x_{\lambda_1}, y_{\lambda_1}) \in H^{\lambda_1}$, let $\lambda_2 = h'(x_{\lambda_1})$, then*

(a) for any $(x_{\lambda_2}, y_{\lambda_2}) \in H^{\lambda_2}$, we have $f(x_{\lambda_2}, y_{\lambda_2}) \leq f(x_{\lambda_1}, y_{\lambda_1})$.

(b) for any $(x_{\lambda_2}, y_{\lambda_2}) \in H^{\lambda_2}$ and $(x_\lambda, y_\lambda) \in H^\lambda$ where $\min\{\lambda_1, \lambda_2\} < \lambda < \max\{\lambda_1, \lambda_2\}$, we have $f(x_{\lambda_2}, y_{\lambda_2}) \leq f(x_\lambda, y_\lambda)$.

(c) for any $(x_{\lambda_2}, y_{\lambda_2}) \in H^{\lambda_2}$, let $\lambda_3 = h'(x_{\lambda_2})$. If $\lambda_2 \geq \lambda_1$, then $\lambda_3 \geq \lambda_2$; otherwise, $\lambda_3 \leq \lambda_2$.

Proof: (a) Due to concavity of h, we have

$$h(x_{\lambda_2}) \leq h(x_{\lambda_1}) + h'(x_{\lambda_1})(x_{\lambda_2} - x_{\lambda_1}) = h(x_{\lambda_1}) + \lambda_2(x_{\lambda_2} - x_{\lambda_1}).$$

Since $(x_{\lambda_2}, y_{\lambda_2}) \in H^{\lambda_2}$, we have $\lambda_2 x_{\lambda_2} + y_{\lambda_2} \leq \lambda_2 x_{\lambda_1} + y_{\lambda_1}$, that is, $\lambda_2(x_{\lambda_2} - x_{\lambda_1}) \leq y_{\lambda_1} - y_{\lambda_2}$. Therefore, $h(x_{\lambda_2}) + y_{\lambda_2} \leq h(x_{\lambda_1}) + y_{\lambda_1}$, that is, $f(x_{\lambda_2}, y_{\lambda_2}) \leq f(x_{\lambda_1}, y_{\lambda_1})$.

(b) Without loss of generality, suppose $\lambda_2 > \lambda_1$. From Lemma 1, we have $x_{\lambda_1} \geq x_\lambda \geq x_{\lambda_2}$ and $y_{\lambda_1} \leq y_\lambda \leq y_{\lambda_2}$. Since h is concave, $h'(x)$ is non-increasing in x and thus $h'(x_\lambda) \geq h'(x_{\lambda_1})$. Therefore, we have

$$h(x_{\lambda_2}) \leq h(x_\lambda) + h'(x_\lambda)(x_{\lambda_2} - x_\lambda) \leq h(x_\lambda) + h'(x_{\lambda_1})(x_{\lambda_2} - x_\lambda) = h(x_\lambda) + \lambda_2(x_{\lambda_2} - x_\lambda).$$

Since $(x_{\lambda_2}, y_{\lambda_2}) \in H^{\lambda_2}$, we have $\lambda_2(x_{\lambda_2} - x_\lambda) \leq y_\lambda - y_{\lambda_2}$. Therefore, $h(x_{\lambda_2}) + y_{\lambda_2} \leq h(x_\lambda) + y_\lambda$, that is, $f(x_{\lambda_2}, y_{\lambda_2}) \leq f(x_\lambda, y_\lambda)$. Using a similar analysis, we also have $f(x_{\lambda_2}, y_{\lambda_2}) \leq f(x_\lambda, y_\lambda)$ when $\lambda_1 > \lambda_2$.

(c) If $\lambda_2 > \lambda_1$, from Lemma 1, we have $x_{\lambda_1} \geq x_{\lambda_2}$. Due to the concavity of h, $\lambda_3 = h'(x_{\lambda_2}) \geq h'(x_{\lambda_1}) = \lambda_2$. If $\lambda_2 > \lambda_1$, similarly we have $\lambda_3 \leq \lambda_2$. □

Part (a) of Proposition 2 indicates that the PSP defined by the gradient of f with respect to $(x_{\lambda_1}, y_{\lambda_1})$ provides a label $(x_{\lambda_2}, y_{\lambda_2})$ superior to $(x_{\lambda_1}, y_{\lambda_1})$. Part (b) of Proposition 2 indicates that optimal solutions (labels) of P^λ with λ in the interval $(\min\{\lambda_1, \lambda_2\}, \max\{\lambda_1, \lambda_2\})$ are inferior to $(x_{\lambda_2}, y_{\lambda_2})$, and therefore this interval can be skipped in the subsequent parametric search. Part (c) of Proposition 2 indicates that such parametric search process always generates a monotonic parameter sequence.

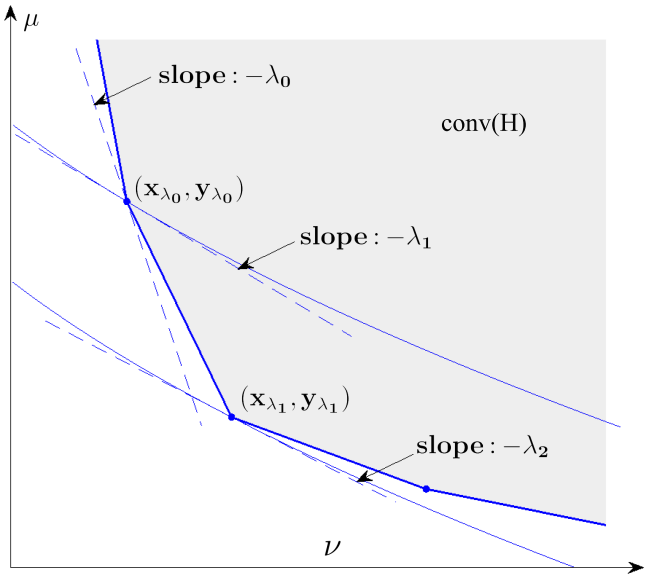

Figure 2 **Illustration of the monotonic descent search.**

The pseudocode of the proposed MDS procedure is given in Algorithm 1. In each iteration, Algorithm 1 either improves the incumbent solution or returns a locally optimal path. Due to the finiteness of the number of extreme non-dominated paths, Algorithm 1 terminates after a finite number of iterations. Figure 2 shows an illustration of Algorithm 1, which starts from λ_0 and returns a locally optimal path with the label $(x_{\lambda_1}, y_{\lambda_1})$ after solving three PSPs.

Algorithm 1 Monotonic descent search: MDS(λ_0)

Input: $\lambda_0 \geq 0$.

Output: $\bar{\lambda} \geq 0$, $(x_{\bar{\lambda}}, y_{\bar{\lambda}}) \in H^{\bar{\lambda}}$, $p^{\bar{\lambda}} \in P^{\bar{\lambda}}$, $(x_{\lambda_0}, y_{\lambda_0}) \in H^{\lambda_0}$ and $p^{\lambda_0} \in P^{\lambda_0}$.

Step 1. Obtain $(x_{\lambda_0}, y_{\lambda_0}) \in H^{\lambda_0}$ and $p^{\lambda_0} \in P^{\lambda_0}$ by solving (P_{λ_0}). Set $k = 1$.

Step 2. Let $\lambda_k = h'(\lambda_{k-1})$. Obtain $(x_{\lambda_k}, y_{\lambda_k}) \in H^{\lambda_k}$ and $p^{\lambda_k} \in P^{\lambda_k}$ by solving (P_{λ_k}).

Step 3. If $\lambda_k = \lambda_{k-1}$, goto step 4; else, set $k = k + 1$ and goto step 2.

Step 4. Return $\bar{\lambda} = \lambda_k$, $(x_{\bar{\lambda}}, y_{\bar{\lambda}}) = (x_{\lambda_k}, y_{\lambda_k})$, $p^{\bar{\lambda}} = p^{\lambda_k}$, $(x_{\lambda_0}, y_{\lambda_0})$ and p^{λ_0}.

3.2. Intersection point search

The MDS procedure may get stuck at locally optimal paths. To address this issue, this subsection proposes an IPS procedure using a bi-directional parametric search strategy. The IPS procedure explores an interval $[\lambda_l, \lambda_u]$ from both the left and right directions, and the objective function value information collected in one direction is used to guide the parametric search in the other direction.

We first introduce related notation. Suppose $\lambda_l \leq \lambda_u$, $(x_{\lambda_l}, y_{\lambda_l}) \in H^{\lambda_l}$ and $(x_{\lambda_u}, y_{\lambda_u}) \in H^{\lambda_u}$. Let the intersection point of two lines $L_l : \lambda_l x + y = R_l$ and $L_u : \lambda_u x + y = R_u$ be (x_c, y_c), where $x_c =$

$\frac{R_u-R_l}{\lambda_u-\lambda_l}$, $y_c=\frac{R_l\lambda_u-R_u\lambda_l}{\lambda_u-\lambda_l}$, $R_l=\lambda_l x_{\lambda_l}+y_{\lambda_l}$ and $R_u=\lambda_u x_{\lambda_u}+y_{\lambda_u}$. From the definitions of $(x_{\lambda_l},y_{\lambda_l})$ and $(x_{\lambda_u},y_{\lambda_u})$ and Lemma 1, it is easy to see that $x_{\lambda_l}\geq x_c\geq x_{\lambda_u}$ and $y_{\lambda_l}\leq y_c\leq y_{\lambda_u}$. Let $f_{up}=\min\{f(x_{\lambda_l},y_{\lambda_l}),f(x_{\lambda_u},y_{\lambda_u})\}$, $f_{low}=\min\{f(x_{\lambda_l},y_{\lambda_l}),f(x_c,y_c),f(x_{\lambda_u},y_{\lambda_u})\}$ and the contour line of $f(x,y)$ at the level f_{up} be $C:f(x,y)=f_{up}$.

The following proposition gives an intersection point property, which shows how to use the intersection points of L_l and L_u, L_l and C and L_u and C to narrow the interval $[\lambda_l,\lambda_u]$.

PROPOSITION 3. *Suppose there exists a label $(\nu^*,\mu^*)\in H^*$, such that $\lambda^*=h'(\nu^*)\in[\lambda_l,\lambda_u]$ and $\nu^*\in[x_{\lambda_u},x_{\lambda_l}]$.*

(a) If $f_{low}\geq f_{up}$, then $f_{up}=f^*$

(b) If $f(x_c,y_c)<f(x_{\lambda_l},y_{\lambda_l})<f(x_{\lambda_u},y_{\lambda_u})$, then the rightmost intersection point (x_m,y_m) of L_u and C satisfying $x_m\in[x_{\lambda_u},x_c]$ exists, and $\lambda^*\in[\lambda_l,h'(x_m)]$ where $h'(x_m)\leq\lambda_u$.

(c) If $f(x_c,y_c)<f(x_{\lambda_u},y_{\lambda_u})<f(x_{\lambda_l},y_{\lambda_l})$, then the leftmost intersection point (x_m,y_m) of L_l and C satisfying $x_m\in[x_c,x_{\lambda_l}]$ exists, and $\lambda^*\in[h'(x_m),\lambda_u]$ where $h'(x_m)\geq\lambda_l$.

Proof: (a) Since $f_{up}\geq f^*$, it is sufficient to show $f_{low}\leq f^*$, which can be proved using the concavity of f and the definitions of H^{λ_l} and H^{λ_u} (Henig 1986).

(b) Let $\delta(x)=h(x)-\lambda_u x+R_u-f_{up}=h(x)-\lambda_u x+R_u-f(x_{\lambda_l},y_{\lambda_l})$. Since $\delta(x_{\lambda_u})=f(x_{\lambda_u},y_{\lambda_u})-f(x_{\lambda_l},y_{\lambda_l})>0$, $\delta(x_c)=f(x_c,y_c)-f(x_{\lambda_l},y_{\lambda_l})<0$ and $\delta(x)$ is continuous, the set of intersection points of L_u and C, such that $x_m\in[x_{\lambda_u},x_c]$, is nonempty. Denote this set as $S=\{(x,y):\delta(x)=0,y=R_u-\lambda_u x,x_{\lambda_u}\leq x\leq x_c\}$, which is a closed and bounded set. Thus, the rightmost intersection point (x_m,y_m), which is the unique optimal solution of the optimization problem $\max\{x:(x,y)\in S\}$, exists.

Since h' is non-increasing, to prove $\lambda^*=h'(\nu^*)\leq h'(x_m)$, it is sufficient to show $\nu^*\geq x_m$. To this end, we show that for any path label $(\nu',\mu')\in H$ satisfying $x_{\lambda_u}\leq\nu'\leq x_m$, we have $f(\nu',\mu')\geq f(x_{\lambda_l},y_{\lambda_l})$. In fact, from the definition of H^{λ_u}, we have $\lambda_u\nu'+\mu'\geq\lambda_u x_{\lambda_u}+y_{\lambda_u}$. Therefore, $f(\nu',\mu')=h(\nu')+\mu'\geq h(\nu')+\widehat{\mu}=f(\nu',\widehat{\mu})$ where $\widehat{\mu}=\lambda_u x_{\lambda_u}+y_{\lambda_u}-\lambda_u\nu'$. Since $\lambda_u\nu'+\widehat{\mu}=\lambda_u x_{\lambda_u}+y_{\lambda_u}$, we have that $(\nu',\widehat{\mu})$ lies on the line segment of L_u between $(x_{\lambda_u},y_{\lambda_u})$ and (x_m,y_m). From the concavity of h, we have $f(\nu',\mu')\geq f(\nu',\widehat{\mu})\geq\min\{f(x_{\lambda_u},y_{\lambda_u}),f(x_m,y_m)\}=f(x_m,y_m)=f(x_{\lambda_l},y_{\lambda_l})$.

Finally, we show $h'(x_m)\leq\lambda_u$ by contradiction. Suppose $h'(x_m)>\lambda_u$, then for any $x\in[x_{\lambda_u},x_m]$, $h'(x)\geq h'(x_m)>\lambda_u$. Thus, $\delta(x_m)=\delta(x_{\lambda_u})+\int_{x_{\lambda_u}}^{x_m}\delta'(x)\mathrm{d}x=\delta(x_{\lambda_u})+\int_{x_{\lambda_u}}^{x_m}(h'(x)-\lambda_u)\mathrm{d}x\geq\delta(x_{\lambda_u})>0$, which contradicts with the fact $\delta(x_m)=0$. Using a similar analysis, we can also prove (c). □

Proposition 3 shows that the gradient of f with respect to the intersection point (x_m,y_m) can also be used to speed up the parametric search. Figure 3 demonstrates the case when $f(x_c,y_c)<f(x_{\lambda_l},y_{\lambda_l})<f(x_{\lambda_u},y_{\lambda_u})$. As shown in the proof of Proposition 3, in this case, there exists an optimal label (ν^*,μ^*), such that $x_m\leq\nu^*\leq x_{\lambda_l}$. Thus, we only need to consider parameters in the interval $[\lambda_l,h'(x_m)]$.

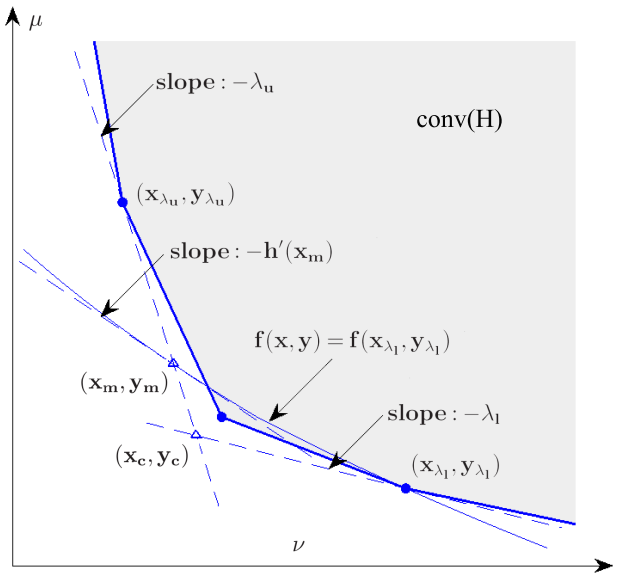

Figure 3 **Illustration of the intersection point search when** $f(x_c, y_c) < f(x_{\lambda_l}, y_{\lambda_l}) < f(x_{\lambda_u}, y_{\lambda_u})$.

To use the intersection point property, we need to calculate intersection points of the curve C and lines L_l or L_u. Such intersection points can be obtained analytically or numerically by a bi-section search. For the MSD-SP problem with $h(x) = \theta\sqrt{x}$ $(\theta \geq 0)$, the rightmost intersection point (x_m, y_m) of L_u and C such that $x_m \in [x_{\lambda_u}, x_c]$ is given as

$$\begin{cases} x_m = \frac{\left(\theta + \sqrt{\theta^2 - 4\lambda_u a_u}\right)^2}{4\lambda_u^2}, \\ y_m = R_u - \lambda_u x_m, \end{cases}$$

where $a_u = f(x_{\lambda_l}, y_{\lambda_l}) - R_u$. The leftmost intersection point (x_m, y_m) of L_l and C such that $x_m \in [x_c, x_{\lambda_l}]$ is given as

$$\begin{cases} x_m = \frac{\left(\theta - \sqrt{\theta^2 - 4\lambda_l a_l}\right)^2}{4\lambda_l^2}, \\ y_m = R_l - \lambda_l x_m, \end{cases}$$

where $a_l = f(x_{\lambda_u}, y_{\lambda_u}) - R_l$.

Algorithm 2 is the pseudocode of the IPS procedure, which calls the MDS procedure as a sub-procedure. Algorithm 2 takes an interval $[\lambda_l, \lambda_u]$ as its input, uses a greedy policy to update the most promising label using the monotonic descent property and then updates the inferior one using the intersection point property. When Algorithm 2 terminates, it either finds an optimal solution of (P1) when $Solved = true$ or returns a narrower interval when $Solved = false$.

The benefits of the proposed IPS procedure are two-fold: (1) the parametric search in one direction can be speeded up using the information collected in the other direction; (2) the IPS procedure terminates only when the paths obtained in both directions are locally optimal and have the same objective function value. As experiments show later, the IPS procedure is able to find optimal paths for all the test instances.

Algorithm 2 Intersection point search: IPS($[\lambda_l, \lambda_u], p^{\lambda_l}, p^{\lambda_u}, (x_{\lambda_l}, y_{\lambda_l}), (x_{\lambda_u}, y_{\lambda_u})$)

Input: $[\lambda_l, \lambda_u]$, $p^{\lambda_l} \in P^{\lambda_l}$, $p^{\lambda_u} \in P^{\lambda_u}$, $(x_{\lambda_l}, y_{\lambda_l}) \in H^{\lambda_l}$ and $(x_{\lambda_u}, y_{\lambda_u}) \in H^{\lambda_u}$.

Output: *Solved*, f_{up}, $\bar{p}$, $[\lambda_l, \lambda_u]$, p^{λ_l}, p^{λ_u}, $(x_{\lambda_l}, y_{\lambda_l})$ and $(x_{\lambda_u}, y_{\lambda_u})$.

Step 1. If $\lambda_u \leq \lambda_l$, then set $Solved = true$ and goto Step 3;

else if $f(x_c, y_c) \geq \min\{f(x_{\lambda_l}, y_{\lambda_l}), f(x_{\lambda_u}, y_{\lambda_u})\}$, then $Solved = true$ and goto Step 3;

else, set $Solved = false$ and goto Step 2.

Step 2. Parametric search based on recursion:

Case 1: $f(x_{\lambda_l}, y_{\lambda_l}) < f(x_{\lambda_u}, y_{\lambda_u})$

If $\lambda_l < h'(x_{\lambda_l})$, then call MDS($h'(x_{\lambda_l})$), and update $\lambda_l = \bar{\lambda}$, $p^{\lambda_l} = p^{\bar{\lambda}}$ and $(x_{\lambda_l}, y_{\lambda_l}) = (x_{\bar{\lambda}}, y_{\bar{\lambda}})$.

Calculate the rightmost intersection point (x_m, y_m) of L_u and C such that $x_m \in [x_{\lambda_u}, x_c]$.

Set $\lambda_u = h'(x_m)$ and obtain $p^{\lambda_u} \in P^{\lambda_u}$ and $(x_{\lambda_u}, y_{\lambda_u}) \in H^{\lambda_u}$ by solving (P_{λ_u}).

Call IPS($[\lambda_l, \lambda_u], p^{\lambda_l}, p^{\lambda_u}, (x_{\lambda_l}, y_{\lambda_l}), (x_{\lambda_u}, y_{\lambda_u})$).

Case 2: $f(x_{\lambda_l}, y_{\lambda_l}) > f(x_{\lambda_u}, y_{\lambda_u})$

If $\lambda_u > h'(x_{\lambda_u})$, then call MDS($h'(x_{\lambda_u})$), and update $\lambda_u = \bar{\lambda}$, $p^{\lambda_u} = p^{\bar{\lambda}}$ and $(x_{\lambda_u}, y_{\lambda_u}) = (x_{\bar{\lambda}}, y_{\bar{\lambda}})$.

Calculate the leftmost intersection point (x_m, y_m) of L_l and C such that $x_m \in [x_c, x_{\lambda_l}]$.

Set $\lambda_l = h'(x_m)$ and obtain $p^{\lambda_l} \in P^{\lambda_l}$ and $(x_{\lambda_l}, y_{\lambda_l}) \in H^{\lambda_l}$ by solving (P_{λ_l}).

Call IPS($[\lambda_l, \lambda_u], p^{\lambda_l}, p^{\lambda_u}, (x_{\lambda_l}, y_{\lambda_l}), (x_{\lambda_u}, y_{\lambda_u})$).

Case 3: $f(x_{\lambda_l}, y_{\lambda_l}) = f(x_{\lambda_u}, y_{\lambda_u})$

If $\lambda_l < h'(x_{\lambda_l})$, then call MDS($h'(x_{\lambda_l})$), and let $\lambda_l = \bar{\lambda}$, $p^{\lambda_l} = p^{\bar{\lambda}}$ and $(x_{\lambda_l}, y_{\lambda_l}) = (x_{\bar{\lambda}}, y_{\bar{\lambda}})$.

If $\lambda_u > h'(x_{\lambda_u})$, then call MDS($h'(x_{\lambda_u})$), and let $\lambda_u = \bar{\lambda}$, $p^{\lambda_u} = p^{\bar{\lambda}}$ and $(x_{\lambda_u}, y_{\lambda_u}) = (x_{\bar{\lambda}}, y_{\bar{\lambda}})$.

If $f(x_{\lambda_l}, y_{\lambda_l}) \neq f(x_{\lambda_u}, y_{\lambda_u})$, then call IPS($[\lambda_l, \lambda_u], p^{\lambda_l}, p^{\lambda_u}, (x_{\lambda_l}, y_{\lambda_l}), (x_{\lambda_u}, y_{\lambda_u})$);

else calculate the intersection point (x_c, y_c) and goto Step 3.

Step 3. If $f(x_{\lambda_l}, y_{\lambda_l}) \leq f(x_{\lambda_u}, y_{\lambda_u})$, then $f_{up} = f(x_{\lambda_l}, y_{\lambda_l})$, $\bar{p} = p^{\lambda_l}$; else, $f_{up} = f(x_{\lambda_u}, y_{\lambda_u})$, $\bar{p} = p^{\lambda_u}$.

If $f_{up} \leq f(x_c, y_c)$ or $\lambda_u \leq \lambda_l$, then set $Solved = true$.

Return *Solved*, f_{up}, $\bar{p}$, $[\lambda_l, \lambda_u]$, p^{λ_l}, p^{λ_u}, $(x_{\lambda_l}, y_{\lambda_l})$ and $(x_{\lambda_u}, y_{\lambda_u})$.

3.3. Interval search algorithm

To guarantee global optimality in theory, we further embed the proposed MDS and IPS procedures into a B&B based interval search algorithm.

A basic version of the interval search algorithm has been proposed in Zhang et al. (2016b). Each branch of the B&B tree is an interval $[\lambda_l, \lambda_u]$ with $(x_{\lambda_l}, y_{\lambda_l}) \in H^{\lambda_l}$ and $(x_{\lambda_u}, y_{\lambda_u}) \in H^{\lambda_u}$. From part (a) of Proposition 3, a lower bound on (P1) over this interval is $f_{low} = \min\{f(x_{\lambda_l}, y_{\lambda_l}), f(x_c, y_c), f(x_{\lambda_u}, y_{\lambda_u})\}$, where (x_c, y_c) is the intersection point of L_l and L_u. If this interval has not been pruned, we divide $[\lambda_l, \lambda_u]$ into two sub-intervals, $[\lambda_l, \lambda_m]$ and $[\lambda_m, \lambda_u]$, where

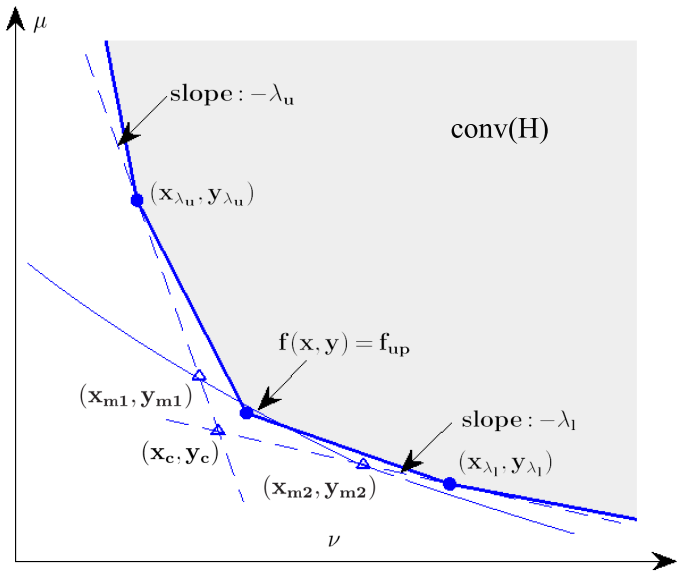

Figure 4 **Illustration of the modified intersection point search when we know an upper bound f_{up} satisfying $f(x_c, y_c) < f_{up} < \min\{f(x_{\lambda_l}, y_{\lambda_l}), f(x_{\lambda_u}, y_{\lambda_u})\}$.**

$\lambda_m = \frac{y_{\lambda_u} - y_{\lambda_l}}{x_{\lambda_l} - x_{\lambda_u}}$, using the perpendicular method; see Dial (1979), Henig (1986) and Chen and Nie (2013).

Algorithm 3 Modified intersection point search: MIPS($f_{up}, \bar{p}, [\lambda_l, \lambda_u], p^{\lambda_l}, p^{\lambda_u}, (x_{\lambda_l}, y_{\lambda_l}), (x_{\lambda_u}, y_{\lambda_u})$)

Input: f_{up}, $\bar{p}$, $[\lambda_l, \lambda_u]$, $p^{\lambda_l} \in P^{\lambda_l}$, $p^{\lambda_u} \in P^{\lambda_u}$, $(x_{\lambda_l}, y_{\lambda_l}) \in H^{\lambda_l}$ and $(x_{\lambda_u}, y_{\lambda_u}) \in H^{\lambda_u}$.

Output: *Solved*, f_{up}, $\bar{p}$, $[\lambda_l, \lambda_u]$, p^{λ_l}, p^{λ_u}, $(x_{\lambda_l}, y_{\lambda_l})$ and $(x_{\lambda_u}, y_{\lambda_u})$.

Step 1. While $f_{up} < \min\{f(x_{\lambda_l}, y_{\lambda_l}), f(x_{\lambda_u}, y_{\lambda_u})\}$

Calculate the intersection point (x_c, y_c) of L_l and L_u.

If $f(x_c, y_c) \geq f_{up}$, then $Solved = true$ and goto Step 3;

Calculate the rightmost intersection point (x_{m1}, y_{m1}) of L_u and C such that $x_{m1} \in [x_{\lambda_u}, x_c]$. Set $\lambda_u = h'(x_{m1})$ and obtain $p^{\lambda_u} \in P^{\lambda_u}$ and $(x_{\lambda_u}, y_{\lambda_u}) \in H^{\lambda_u}$ by solving (P_{λ_u}).

Calculate the leftmost intersection point (x_{m2}, y_{m2}) of L_l and C such that $x_{m2} \in [x_c, x_{\lambda_l}]$.

Set $\lambda_l = h'(x_{m2})$ and obtain $p^{\lambda_l} \in P^{\lambda_l}$ and $(x_{\lambda_l}, y_{\lambda_l}) \in H^{\lambda_l}$ by solving (P_{λ_l}).

Step 2. Call IPS($[\lambda_l, \lambda_u], p^{\lambda_l}, p^{\lambda_u}, (x_{\lambda_l}, y_{\lambda_l}), (x_{\lambda_u}, y_{\lambda_u})$).

Step 3. Return *Solved*, f_{up}, $\bar{p}$, $[\lambda_l, \lambda_u]$, p^{λ_l}, p^{λ_u}, $(x_{\lambda_l}, y_{\lambda_l})$ and $(x_{\lambda_u}, y_{\lambda_u})$.

To improve the interval search algorithm, we first observe that the intersection point property can be further extended to the case when an upper bound f_{up} on (P1) satisfying $f(x_c, y_c) < f_{up} < \min\{f(x_{\lambda_l}, y_{\lambda_l}), f(x_{\lambda_u}, y_{\lambda_u})\}$ is known. Specifically, using a similar proof as that for Proposition 3, we have that if there exists a label $(\nu^*, \mu^*) \in H^*$, such that $\lambda^* = h'(\nu^*) \in [\lambda_l, \lambda_u]$ and $\nu^* \in [x_{\lambda_u}, x_{\lambda_l}]$,

then $x_{m1} \leq \nu^* \leq x_{m2}$ and $h'(x_{m2}) \leq \lambda^* \leq h'(x_{m1})$, where (x_{m1}, y_{m1}) is the rightmost intersection point of L_u and C such that $x_{m1} \in [x_{\lambda_u}, x_c]$, and (x_{m2}, y_{m2}) is the leftmost intersection point of L_l and C such that $x_{m2} \in [x_c, x_{\lambda_l}]$; see Figure 4. This observation enables us to use the incumbent solution to speed up the parametric search in each interval. Based on this observation, we propose a modified IPS procedure in Algorithm 3, which either prunes an interval or returns a narrower interval.

The pseudocode of the interval search algorithm is given in Algorithm 4. In Step 2, the classical best-first, depth-first and width-first strategies can be used to select the next interval. In Step 3, Algorithm 4 divides the selected interval into two sub-intervals and calls the MIPS procedure to narrow each sub-interval using the incumbent objective function value f_{up}. Then, f_{up} is updated and sub-intervals are added to I if not pruned. Due to the finiteness of the number of paths, Algorithm 4 returns an optimal path after a finite number of iterations.

Algorithm 4 Interval search algorithm

Step 1. Calculate an initial parameter interval $[\lambda_L, \lambda_U]$.

Obtain $p^{\lambda_L} \in P^{\lambda_L}$ and $(x_{\lambda_L}, y_{\lambda_L}) \in H^{\lambda_L}$ by solving (P_{λ_L}).

Obtain $p^{\lambda_U} \in P^{\lambda_U}$ and $(x_{\lambda_U}, y_{\lambda_U}) \in H^{\lambda_U}$ by solving (P_{λ_U}).

Call IPS$([\lambda_L, \lambda_U], p^{\lambda_L}, p^{\lambda_U}, (x_{\lambda_L}, y_{\lambda_L}), (x_{\lambda_U}, y_{\lambda_U}))$ to get *Solved*, f_{up}, $\bar{p}$, $[\lambda_l, \lambda_u]$, p^{λ_l}, p^{λ_u}, $(x_{\lambda_l}, y_{\lambda_l})$ and $(x_{\lambda_u}, y_{\lambda_u})$.

If $Solved = true$, goto Step 4; else, set $I = \{[\lambda_l, \lambda_u]\}$.

Step 2. If $I = \emptyset$, goto Step 4; else, select and remove an interval $[\lambda_l, \lambda_u]$ from I.

Step 3. Set $\lambda_m = \frac{y_{\lambda_u} - y_{\lambda_l}}{x_{\lambda_l} - x_{\lambda_u}}$ and obtain $p^{\lambda_m} \in P^{\lambda_m}$ and $(x_{\lambda_m}, y_{\lambda_m}) \in H^{\lambda_m}$ by solving (P_{λ_m}).

Call MIPS$(f_{up}, \bar{p}, [\lambda_l, \lambda_m], p^{\lambda_l}, p^{\lambda_m}, (x_{\lambda_l}, y_{\lambda_l}), (x_{\lambda_m}, y_{\lambda_m}))$ to get *Solved*, f'_{up}, $\bar{p}'$, $[\lambda'_l, \lambda'_u]$, $p^{\lambda'_l}$, $p^{\lambda'_u}$, $(x_{\lambda'_l}, y_{\lambda'_l})$ and $(x_{\lambda'_u}, y_{\lambda'_u})$.

If $f'_{up} < f_{up}$, then $f_{up} = f'_{up}$ and $\bar{p} = \bar{p}'$. If $Sovled = false$, set $I = I \cup \{[\lambda'_l, \lambda'_u]\}$.

Call MIPS$(f_{up}, \bar{p}, [\lambda_m, \lambda_u], p^{\lambda_m}, p^{\lambda_u}, (x_{\lambda_m}, y_{\lambda_m}), (x_{\lambda_u}, y_{\lambda_u}))$ to get *Solved*, f'_{up}, $\bar{p}'$, $[\lambda'_l, \lambda'_u]$, $p^{\lambda'_l}$, $p^{\lambda'_u}$, $(x_{\lambda'_l}, y_{\lambda'_l})$ and $(x_{\lambda'_u}, y_{\lambda'_u})$.

If $f'_{up} < f_{up}$, then $f_{up} = f'_{up}$ and $\bar{p} = \bar{p}'$. If $Sovled = false$, set $I = I \cup \{[\lambda'_l, \lambda'_u]\}$.

Goto Step 2.

Step 4. Return f_{up} and $\bar{p}$.

4. A speedup label correcting algorithm

In the parametric search, a series of PSPs need to be solved. These PSPs are weighted shortest path problems with similar cost structures in the same network. This section proposes a speedup

LC algorithm, which utilizes information collected in the process of solving previous PSPs to speed up the solving process of subsequent PSPs.

The classical LC algorithm solves shortest path problems with arbitrary arc cost c_{ij} $(ij \in A)$ but no negative cycle. The LC algorithm maintains a list of nodes V, called the candidate node list, and a distance label d_i for each node i, which represents an upper bound on the shortest distance from s to i. It progressively discovers shorter paths from s to every node $i \in N$ by iteratively updating distance labels. Specifically, the LC algorithm initializes $V = \{s\}$, $d_s = 0$ and $d_i = +\infty$ for $i \in N \setminus \{s\}$. Then, it selects and removes a node i from V. For each $j \in N$ satisfying both $ij \in A$ and $d_i + c_{ij} < d_j$, it corrects the distance label d_j by $d_i + c_{ij}$ and sets $V = V \cup \{j\}$. When $V = \emptyset$, the LC algorithm terminates and returns d_i for each $i \in N$, which is the shortest distance from s to i. The computational performance of the LC algorithm is mainly determined by the number of labeling operations. In the following, we show how to reduce the number of labeling operations using information collected in the process of solving related shortest path problems.

Consider a parameterized network $G^{\lambda_k}(N, A)$, where the arc cost of $ij \in A$ is given as $\lambda_k \nu_{ij} + \mu_{ij}$. Let p_{it}^k be a shortest path from i to t in $G(\lambda_k)$. The label of p_{it}^k is given as $(\nu(p_{it}^k), \mu(p_{it}^k))$, where $\nu(p_{it}^k) = \sum_{lj \in p_{it}^k} \nu_{lj}$ and $\mu(p_{it}^k) = \sum_{lj \in p_{it}^k} \mu_{lj}$. The shortest distance from i to t in $G^{\lambda_k}(N, A)$ is denoted as $d_{it}^{\lambda_k} = \lambda_k \nu(p_{it}^k) + \mu(p_{it}^k)$. The following proposition shows how to use shortest paths in two "neighbourhood" parameterized networks $G^{\lambda_1}(N, A)$ and $G^{\lambda_2}(N, A)$ to estimate upper and lower bounds on d_{it}^{λ} for any $\lambda \in [\lambda_1, \lambda_2]$.

PROPOSITION 4. *For any $\lambda \in [\lambda_1, \lambda_2]$ and $i \in N$, we have*

$$\lambda \nu_i + \mu_i \le d_{it}^{\lambda} \le \min\{\lambda \nu(p_{it}^1) + \mu(p_{it}^1), \lambda \nu(p_{it}^2) + \mu(p_{it}^2)\},$$

where $\nu_i = \frac{d_{it}^{\lambda_2} - d_{it}^{\lambda_1}}{\lambda_2 - \lambda_1}$ and $\mu_i = \frac{\lambda_2 d_{it}^{\lambda_1} - \lambda_1 d_{it}^{\lambda_2}}{\lambda_2 - \lambda_1}$. Furthermore, if $\lambda_3 \le \lambda_1 \le \lambda \le \lambda_2 \le \lambda_4$, then

$$\min\{\lambda \nu(p_{it}^1) + \mu(p_{it}^1), \lambda \nu(p_{it}^2) + \mu(p_{it}^2)\} \le \min\{\lambda \nu(p_{it}^3) + \mu(p_{it}^3), \lambda \nu(p_{it}^4) + \mu(p_{it}^4)\}, \tag{1}$$

$$\lambda \nu_i + \mu_i \ge \lambda \nu_i' + \mu_i', \tag{2}$$

where $\nu_i' = \frac{d_{it}^{\lambda_4} - d_{it}^{\lambda_3}}{\lambda_4 - \lambda_3}$ and $\mu_i' = \frac{\lambda_4 d_{it}^{\lambda_3} - \lambda_3 d_{it}^{\lambda_4}}{\lambda_4 - \lambda_3}$.

Proof: Since both p_{it}^1 and p_{it}^2 are valid paths from i to t, it is clear that $\min\{\lambda \nu(p_{it}^1) + \mu(p_{it}^1), \lambda \nu(p_{it}^2) + \mu(p_{it}^2)\}$ is an upper bound on d_{it}^{λ}. For any path $p_{it} \in P_{it}$ with a label $(\nu(p_{it}), \mu(p_{it}))$, from the optimality of p_{it}^1 and p_{it}^2, we have $\lambda_1 \nu(p_{it}) + \mu(p_{it}) \ge d_{it}^{\lambda_1}$ and $\lambda_2 \nu(p_{it}) + \mu(p_{it}) \ge d_{it}^{\lambda_2}$. Therefore, a lower bound on d_{it}^{λ} can be given by the optimal value of the linear programming problem: $\min\{\lambda x + y : \lambda_1 x + y \ge d_{it}^{\lambda_1}, \lambda_2 x + y \ge d_{it}^{\lambda_2}, (x, y) \in R^2\}$. Since $\lambda_1 \le \lambda \le \lambda_2$, an optimal solution of this problem is given as $(x^*, y^*) = (\nu_i, \mu_i)$ and thus $\lambda \nu_i + \mu_i \le d_{it}^{\lambda}$.

To prove (1), it is sufficient to show $\lambda\nu(p_{it}^1)+\mu(p_{it}^1)\le\lambda\nu(p_{it}^3)+\mu(p_{it}^3)$ and $\lambda\nu(p_{it}^2)+\mu(p_{it}^2)\le\lambda\nu(p_{it}^4)+\mu(p_{it}^4)$. From the optimality of p_{it}^1, we have $\lambda_1\nu(p_{it}^1)+\mu(p_{it}^1)\le\lambda_1\nu(p_{it}^3)+\mu(p_{it}^3)$, that is, $\mu(p_{it}^1)-\mu(p_{it}^3)\le\lambda_1(\nu(p_{it}^3)-\nu(p_{it}^1))$. Since $\lambda_3\le\lambda_1\le\lambda$, from Lemma 1, we have $\nu(p_{it}^3)\ge\nu(p_{it}^1)$ and thus $\mu(p_{it}^1)-\mu(p_{it}^3)\le\lambda(\nu(p_{it}^3)-\nu(p_{it}^1))$, that is, $\lambda\nu(p_{it}^1)+\mu(p_{it}^1)\le\lambda\nu(p_{it}^3)+\mu(p_{it}^3)$. Similarly, we can prove $\lambda\nu(p_{it}^2)+\mu(p_{it}^2)\le\lambda\nu(p_{it}^4)+\mu(p_{it}^4)$.

Note that $\lambda\nu_i'+\mu_i'=\min\{\lambda x+y:\lambda_3x+y\ge d_{it}^{\lambda_3},\lambda_4x+y\ge d_{it}^{\lambda_4},(x,y)\in R^2\}$. Thus, to prove (2), it is sufficient to show that (ν_i,μ_i) is a feasible solution of this minimization problem. Since $\lambda_1\nu_i+\mu_i=d_{it}^{\lambda_1}$, $\lambda_1\ge\lambda_3$ and $\nu(p_{it}^1)\ge\nu_i$, we have $\mu_i-\mu(p_{it}^1)=\lambda_1(\nu(p_{it}^1)-\nu_i)\ge\lambda_3(\nu(p_{it}^1)-\nu_i)$, that is, $\lambda_3\nu_i+\mu_i\ge\lambda_3\nu(p_{it}^1)+\mu(p_{it}^1)$. Due to the optimality of p_{it}^3, we have $\lambda_3\nu_i+\mu_i\ge\lambda_3\nu(p_{it}^1)+\mu(p_{it}^1)\ge d_{it}^{\lambda_3}$. Similarly, we can prove $\lambda_4\nu_i+\mu_i\ge d_{it}^{\lambda_4}$. □

To use Proposition 4 to reduce the number of labeling operations at node i, we need both p_{it}^1 and p_{it}^2 in advance.For this purpose, in the initialization phase, i.e., Step 1 of the interval search algorithm, we use the classical LC algorithm to solve two single-origin all-destinations shortest path problems in the reversed networks $\bar{G}^{\lambda_1}(N,\bar{A})$ and $\bar{G}^{\lambda_2}(N,\bar{A})$, where $\lambda_1=\lambda_L$, $\lambda_2=\lambda_U$ and $\bar{A}=\{ji:i\in N,j\in N,ij\in A\}$. Specifically, solving the single-origin all-destinations shortest path problem from t to all other nodes in $\bar{G}^{\lambda_k}(N,\bar{A})$ gives a shortest path p_{it}^k from any node i to t in $\bar{G}^{\lambda_k}(N,\bar{A})$ $(k=1,2)$. Therefore, the initialization phase provides the required paths p_{it}^k $(k=1,2)$ from any node i to t for estimating upper and lower bounds on d_{it}^{λ} for any $\lambda\in[\lambda_L,\lambda_U]$.

The pseudocode of a basic version of the speedup LC algorithm is given in Algorithm 5. Algorithm 5 first estimates a lower bound $\tilde{d}_{it}^{\lambda}$ on d_{it}^{λ} for each $i\in N$ and an upper bound $\bar{d}_{st}^{\lambda}$ on d_{st}^{λ}. In Step 2, it uses the first-in first-out rule to select the next node. Compared to the classical LC algorithm, the label of node j gets corrected only when both $d_i+\lambda\nu_{ij}+\mu_{ij}<d_j$ and $d_i+\lambda\nu_{ij}+\mu_{ij}+\tilde{d}_{jt}^{\lambda}<\bar{d}_{st}^{\lambda}$ are satisfied. The following proposition 5 validates the correctness of Algorithm 5.

PROPOSITION 5. *If $G^{\lambda}(N,A)$ contains no negative cycle that is reachable from s, Algorithm 5 finds a shortest path $p\in P^{\lambda}$ in $\mathcal{O}(nm)$ time.*

Proof: We use the concept of pass over the queue V to show the correctness of Algorithm 5. Pass one consists of iterations in Step 3 applied to the node s. The subsequent pass consists of iterations in Step 3 applied to these nodes that entered V during the preceding pass. Since there is no negative cycle, Steps 2 and 3 terminate after implementing at most n passes over the queue V. If $d_{st}^{\lambda}=\lambda\nu(p_{st}^{k^*})+\mu(p_{st}^{k^*})$, Algorithm 5 finds a shortest path from s to t in $G^{\lambda}(N,A)$ in Step 1; otherwise, Algorithm 5 finds a shortest path in Steps 2 and 3. Note that each pass requires at most $O(m)$ operations. Therefore, Algorithm 5 finds a shortest path $p\in P^{\lambda}$ in $\mathcal{O}(nm)$ time. □

The speedup LC algorithm can be further improved as follows. First, the second part of Proposition 4 shows that tighter lower and upper bounds on d_{it}^{λ} can be obtained using shortest paths in

Algorithm 5 Speedup LC algorithm

Input: λ, λ_k, p_{st}^k, $(\nu(p_{it}^k), \mu(p_{it}^k)$ for any $i \in N$ and $k = 1, 2$

Output: $p \in P^\lambda$

Step 1. Initialize the queue $V = \{s\}$. Set $pred[i] = Null$, $d_i = +\infty$ for $i \in N$ and $d_s = 0$.

For each $i \in N$, calculate (ν_i, μ_i) and set $\tilde{d}_{it}^\lambda = \lambda\nu_i + \mu_i$.

Set $\bar{d}_{st}^\lambda = \lambda\nu(p_{st}^{k^*}) + \mu(p_{st}^{k^*})$ and $p = p_{st}^{k^*}$ where $k^* = \arg\min\{\lambda\nu(p_{st}^1) + \mu(p_{st}^1) : k = 1, 2\}$.

Initialize $pred[i]$ and d_i for each node i on path p^{k^*}. If $\bar{d}_{st}^\lambda \le \tilde{d}_{st}^\lambda$, then goto Step 4.

Step 2. If $V = \emptyset$, goto Step 4; else, select and remove a node i from V.

Step 3. For each $j \in N$ such that $ij \in A$

If $d_i + \lambda\nu_{ij} + \mu_{ij} < d_j$ and $d_i + \lambda\nu_{ij} + \mu_{ij} + \tilde{d}_{jt}^\lambda < \bar{d}_{st}^\lambda$

Set $d_j = d_i + \lambda\nu_{ij} + \mu_{ij}$ and set $pred[j] = i$.

If $j \neq t$, then set $V = V \cup \{j\}$;

else, set $\bar{d}_{st}^\lambda = d_j$ and if $\bar{d}_{st}^\lambda \le \tilde{d}_{st}^\lambda$, then goto Step 4.

Goto Step 2.

Step 4. Return the path p by tracing back $pred[t]$.

the "nearest neighbour" networks. Thus, in Step 1, for each $i \in N$, known shortest paths from i to t in the "nearest neighbour" networks should be used to calculate $\tilde{d}_{it}^\lambda$ and $\bar{d}_{st}^\lambda$. Second, since the attributes on each arc are nonnegative, then in Step 1, the node $i \in N \setminus \{s\}$ and its associated arcs satisfying $\tilde{d}_{it}^\lambda \ge \bar{d}_{st}^\lambda$ can be removed from the network.

Although the proposed speedup LC Algorithm has the same worst-case computational complexity as the classical LC algorithm, our experiments show that the number of labeling operations can be greatly reduced and the run time can be speeded up by orders of magnitude.

5. Numerical experiments

In this section, we test the parametric search (PS) method on both real transportation networks and grid networks by comparison with most recently proposed algorithms. Specifically, we first compare the performance of the PS method with the improved IL (IIL) algorithm (Khani and Boyles 2015) and the OA algorithm (Shahabi et al. 2013, Shahabi and Boyles 2015) for the MSD-SP problem on both real transportation networks and grid networks. Then, we test the performance of the PS method for the route choice problem with a quadratic disutility function (RC-QDF) on grid networks by comparison with the IIL algorithm. We describe the experimental setup in Subsection 5.1 and report experimental results for the MSD-SP problem and the RC-QDF problem in Subsections 5.2 and 5.3, respectively.

5.1. Experimental setup

5.1.1. Implementation details of algorithms The depth-first strategy is used to select the next interval in Step 2 of Algorithm 4. In LC algorithms, we use an adjacency list to represent the network; see Chapter 22 in Thomas et al. (2001) for details. The first-in first-out queue V is implemented by a *circular* array Q of size n. Q has a *head* and a *tail*, such that its elements reside in locations $Q[head]$, $Q[head+1], \cdots, Q[tail-1]$, where we "wrap around" in the sense that location 1 immediately follows location n in a circular order. We also create a *boolean* array $Flag$ of size n, such that for each $i \in N$, if i is an element of Q, $Flag[i] = 1$; otherwise, $Flag[i] = 0$. In Step 2 of Algorithm 5, we select and delete the node i at the head of Q, i.e., $i = Q[head]$, set $Flag[i] = 0$ and update

$$head = \begin{cases} head + 1, & \text{if } head < n, \\ 1, & \text{if } head = n. \end{cases}$$

In Step 3 of Algorithm 5, a node i is inserted into $Q[tail]$ only when $Flag[i] = 0$, and after this insertion, we set $Flag[i] = 1$ and update

$$tail = \begin{cases} tail + 1, & \text{if } tail < n, \\ 1, & \text{if } tail = n. \end{cases}$$

With the aid of the *circular* array Q and the *boolean* array $Flag$, it only takes $\mathcal{O}(1)$ time to insert a node into Q or delete a node from Q. The IIL algorithm is coded based on the pseudocode of Algorithm 2 in Khani and Boyles (2015). Master problems of the OA algorithm are modeled by YALMIP (Löfberg 2004) and solved by CPLEX 12.6. All algorithms are coded using Matlab R2013a and tested on a 64 bit PC with an Intel Core i5-4570 CPU and 8 GB RAM.

5.1.2. Test problems The MSD-SP problem is of the form: $\min\left\{\theta\sqrt{\nu(p)} + \mu(p) : p \in P\right\}$, where $\theta > 0$, $\mu(p)$ and $\nu(p)$ represent the mean and variance of the travel time of a path p. The RC-QDF problem has been considered by Chen and Nie (2013), Wu and Nie (2011). We test the RC-QDF problem of the form: $\min\left\{\theta\nu(p)\frac{a-\nu(p)}{b} + \mu(p) : p \in P\right\}$, where $\theta, a, b \geq 0$, and $\mu(p)$ and $\nu(p)$ represent two attributes of a path p. To balance the impact of both attributes on the objective function, we set $a = 2\nu_{\max}$ and $b = 2\nu_{\max} - \nu_{\min}$, such that $0 \leq \frac{a-\nu(p)}{b} \leq 1$ for any $p \in P$, where $\nu_{\max}$ and $\nu_{\min}$ are upper and lower bounds on $\nu(p)$ over $p \in P$. For both MSD-SP and RC-QDF problems, the value of θ varies from 0.1 to 10 to test its impact on computational performance of different algorithms.

5.1.3. Data sets Five real transportation networks are considered, including two small size networks in Anaheim and Chicago Sketch, and three large size networks in Chicago Regional, Philadelphia and Birmingham City. Data sets of these networks are obtained from Bar-Gera (2016). For each arc of the test transportation networks, its expected travel time is given by Bar-Gera

Table 1 Real transportation networks

Networks	n	m
Anaheim	416	914
Chicago Sketch	933	2,950
Chicago Regional	12,982	39,018
Philadelphia	13,389	40,003
Birmingham City	14,639	33,937

Table 2 Grid networks

Networks	$w \times h$	n	m
G1	20×50	1,002	3,960
G2	32×32	1,026	4,032
G3	50×20	1,002	3,900
G4	20×500	10,002	39,960
G5	100×100	10,002	39,800
G6	500×20	10,002	39,000
G7	$20 \times 5,000$	100,002	399,960
G8	320×320	102,402	408,960
G9	$5,000 \times 20$	100,002	390,000

(2016) and the standard deviation of its travel time is generated randomly within 0.15 of its expected travel time. More details of these networks are given in Table 1.

We also test the proposed method on grid networks, which have been widely used to test bicriterion shortest path algorithms in Matthew and Kevin (2005), Raith and Ehrgott (2009), Chen and Nie (2013). Besides an origin node s and a destination node t, other nodes of a grid network are arranged in a rectangular grid with a width w and a height h. Every node has at most four outgoing arcs (up, down, left and right) to its immediate neighbours. The grid network with a height h and a width w has $n = 2 + wh$ nodes and $m = 4wh - 2w$ arcs. To test how the structure of grid networks impacts the performance of different algorithms, we generate three groups of grid networks. The grid networks within each group have a similar number of nodes and arcs, but different values of w and h. Table 2 gives the network structure of these grid networks. Both attributes of each arc are generated uniformly from $\{1, 2, \cdots, 1000\}$.

In the experiments, for each test network of given size, we construct 20 instances with randomly generated attributes and origin-destination pairs, and report the average performance of different algorithms.

5.2. Results for the MSD-SP problem

5.2.1. Real transportation networks

We report computational results of the proposed PS method for the MSD-SP problem on the real transportation networks with the value of θ varying from 0.1 to 10 in Tables 3 and 4. PS1 and PS2 denote the PS methods using the classical LC algorithm and the speedup LC algorithm,

Table 3 Performance of PS1 for the MSD-SP problem on the real transportation networks

		PS1			Improvement	
Networks	θ	CPU (s)	PSP	Label	IIL (%)	OA (%)
Anaheim	0.1	0.001	3.8	4,046	95.229	99.924
	1	0.001	4.0	4,161	93.388	99.899
	10	0.001	4.9	3,685	93.800	99.949
Chicago Sketch	0.1	0.005	4.0	8,139	95.215	99.537
	1	0.004	4.0	11,392	96.523	99.691
	10	0.004	5.6	15,892	95.510	99.984
Chicago Regional	0.1	0.180	3.8	625,099	97.012	98.326
	1	0.312	4.3	1,083,221	94.963	98.538
	10	0.476	6.0	1,656,838	92.194	99.893
Philadelphia	0.1	0.225	3.2	752,274	95.879	98.327
	1	0.388	4.1	1,348,254	93.257	97.186
	10	0.506	5.3	1,807,048	90.772	99.748
Birmingham City	0.1	0.346	3.6	1,351,374	92.330	96.779
	1	0.591	4.4	2,276,532	87.273	96.387
	10	0.697	4.9	2,753,872	85.346	99.700

Table 4 Performance of PS2 for the MSD-SP problem on real transportation networks

		PS2			Improvement	
Networks	θ	CPU (s)	PSP	Label	IIL (%)	OA (%)
Anaheim	0.1	0.002	3.8	3,591	89.066	99.826
	1	0.002	4.0	3,700	90.496	99.854
	10	0.001	4.9	1,587	93.800	99.949
Chicago Sketch	0.1	0.010	4.0	6,481	89.941	99.028
	1	0.004	4.0	9,298	96.429	99.028
	10	0.004	5.6	10,440	96.429	99.987
Chicago Regional	0.1	0.085	3.8	252,189	98.600	99.216
	1	0.148	4.3	500,673	97.613	99.307
	10	0.176	6.0	585,673	97.115	99.960
Philadelphia	0.1	0.153	3.2	475,869	97.211	98.868
	1	0.267	4.1	901,216	95.353	98.060
	10	0.285	5.3	1,002,827	94.798	99.858
Birmingham City	0.1	0.157	3.6	575,126	96.527	98.542
	1	0.297	4.4	1,147,640	93.607	98.186
	10	0.324	4.9	1,254,918	93.196	99.860

respectively. The third to fifth columns give the average CPU time, the number of PSPs and the total number of labeling operations. The last two columns give the improvement in CPU time made by PS1 and PS2 in comparison with the IIL and OA algorithms. From Tables 3 and 4, we have the following observations. First, both PS1 and PS2 reduce the computation time of the IIL and OA algorithms by one to two orders of magnitude. Second, the average CPU time and the total number of labeling operations of PS1 can be greatly reduced by using the speedup LC algorithm, especially for large size problems. Third, for all the test instances, only a very small number (no more than 10) of PSPs need to be solved by both PS1 and PS2. Finally, the impact of θ on the performance

Table 5 Performance of the speedup LC algorithm for the MSD-SP problem on real transportation networks

		LC		speedup LC		Improvement (%)	
Networks	θ	CPU (ms)	Label	CPU (ms)	Label	CPU	Label
Anaheim	0.1	0.141	526	0.025	9	82.270	98.377
	1	0.141	658	0.067	13	52.719	97.965
	10	0.184	924	0.092	68	50.181	92.626
Chicago Sketch	0.1	0.618	1,803	0.200	15	67.638	99.187
	1	0.500	1,837	0.050	18	90.000	98.999
	10	0.442	2,102	0.115	132	73.962	93.723
Chicago Regional	0.1	38.050	133,723	0.550	23	98.555	99.983
	1	45.333	156,835	0.517	95	98.860	99.939
	10	58.729	207,019	0.668	770	98.862	99.628
Philadelphia	0.1	38.200	127,256	0.200	41	99.476	99.968
	1	66.667	229,889	0.567	93	99.150	99.960
	10	77.940	274,998	0.700	1,030	99.102	99.625
Birmingham City	0.1	71.325	276,703	0.400	54	99.439	99.980
	1	88.167	340,243	0.517	138	99.414	99.960
	10	107.163	420,447	0.723	1,967	99.325	99.532

of the proposed PS method is relatively small. For example, when the value of θ increases by a factor of 100, the average CPU time of both PS1 and PS2 for all test networks only increases by a factor of no more than 2.5.

Table 5 gives a detailed comparison of the proposed speedup LC algorithm and the classical one. Note that in order to use the speedup LC algorithm, PS2 needs to solve two single-origin all-destinations shortest path problems in the initialization phase. Thus, computational results reported in Table 5 are calculated based on the computation for PSPs after the initialization phase. In particular, the third to sixth columns of Table 5 give the average CPU time, and the average numbers of labeling operations of both the classical and speedup LC algorithms for a PSP after the initialization phase. The last two columns of Table 5 give the improvement in both CPU time and the number of labeling operations made by the speedup LC algorithm. Orders of magnitude improvement is observed, especially for large size problems. Since only no more than 10 PSPs need to be solved by PS2, the efficiency of PS2 using the speedup LC algorithm is mainly determined by the computation time for the first two shortest path problems. Therefore, the computation time of PS2 can be approximated by $O((2+\epsilon)nm)$, where $0 \leq \epsilon \leq 1$.

In addition to experimental results reported in Tables 3, 4 and 5, the IPS procedure finds optimal paths for all the test instances and no B&B procedure is required.

5.2.2. Grid networks

We proceed to test the performance of the PS method on grid networks. Table 6 reports the average performance of PS1 and PS2 for the MSD-SP problem on grid networks based on 20 randomly generated instances when $\theta = 1$. The fifth and ninth columns of Table 6 give the improvement in

Table 6 Performance of PS1 and PS2 for the MSD-SP problem on grid networks

	PS1				PS2			
	CPU (s)	PSP	Label	Improv (%)	CPU (s)	PSP	Label	Improv (%)
G1	0.003	2.9	8,644	97.735	0.002	2.9	5,868	97.910
G2	0.004	3.4	13,476	97.743	0.002	3.4	7,151	98.558
G3	0.005	3.4	18,549	97.331	0.004	3.4	10,932	97.973
G4	0.018	2.4	69,610	99.332	0.019	2.4	58,844	99.273
G5	0.080	3.4	315,616	98.509	0.057	3.4	193,750	98.943
G6	0.529	4.1	2,115,137	99.853	0.258	4.1	1,032,817	98.378
G7	0.235	2.2	646,702	99.853	0.235	2.2	585,993	99.853
G8	4.284	3.8	12,525,145	98.657	2.210	3.8	6,363,157	99.307
G9	61.022	4.0	219,017,852	96.269	32.063	4.0	112,106,563	98.040

Table 7 Performance of the speedup LC algorithm for the MSD-SP problem on grid networks

	LC		speedup LC		Improvement (%)	
	CPU (ms)	Label	CPU (ms)	Label	CPU	Label
G1	0.450	1,579	0.049	8	88.889	99.519
G2	0.850	3,297	0.050	12	94.118	99.638
G3	1.100	3,883	0.050	16	95.455	99.579
G4	2.100	8,835	0.100	10	95.238	99.887
G5	15.883	65,373	0.367	39	97.691	99.940
G6	132.033	520,807	0.717	517	99.457	99.901
G7	20.900	58,696	0.300	20	98.565	99.966
G8	1,032.500	3,003,600	3.750	375	99.637	99.988
G9	15,176.400	54,661,882	7.150	8,451	99.953	99.985

computation time made by PS1 and PS2 in comparison with the IIL algorithm. Comparison with the OA algorithm is not reported here because the IIL algorithm outperforms the OA algorithm. The first three observations from Table 3 and 4 are still valid for the grid networks, and a greater degree of improvement in computation time is observed. Table 6 further shows that the network structure has a significant impact on both the PS method and the IIL algorithm. However, the increase in the number of PSPs due to the adverse network structure is still relatively small.

Computational performance of Nikolova (2009)'s parametric methods for the MSD-SP problem on grid networks with a similar experimental setup has been reported in Nikolova (2009). Nikolova (2009)'s exact PS method needs to solve more than 50 PSPs to find the optimal path in a grid network with $h = w = 250$ while on average, our PS method only needs to solve 3.8 PSPs in a grid network with $h = w = 320$.

Table 7 gives computational details of the classical and speedup LC algorithms used by PS1 and PS2. We also observe that the speedup LC algorithm reduces the number of labeling operations by more than two orders of magnitude for all the test instances. Another observation is that the speedup LC algorithm has a slower increase in computation time as the network structure becomes more adverse. For example, when the test network varies from G1, G4 and G7 to G3, G6 and G9, the CPU time of the classical LC algorithm increases by 144%, 6,187% and 72,514% while

Table 8 Performance of PS1 and PS2 for the RC-QDF problem on grid networks

		PS1				PS2			
	θ	CPU (s)	PSP	Label	Improv (%)	CPU (s)	PSP	Label	Improv (%)
G1	0.1	0.004	4.0	10,745	97.283	0.003	4.0	5,381	97.585
	1	0.003	4.1	8,446	97.330	0.003	4.1	4,151	97.788
	10	0.004	4.1	10,679	97.276	0.003	4.1	5,481	97.588
G2	0.1	0.006	4.0	14,930	96.701	0.003	4.0	6,501	98.237
	1	0.004	4.2	10,652	97.746	0.003	4.2	5,580	98.324
	10	0.005	4.0	14,704	97.115	0.004	4.0	7,404	97.964
G3	0.1	0.006	4.0	19,499	97.132	0.004	4.0	9,630	98.058
	1	0.004	4.3	13,164	98.045	0.003	4.3	6,365	98.610
	10	0.007	4.0	21,388	96.922	0.004	4.0	9,496	98.109
G4	0.1	0.035	4.0	106,111	98.808	0.026	4.0	53,630	99.103
	1	0.027	4.1	85,867	99.067	0.025	4.1	42,206	99.143
	10	0.036	4.0	106,611	98.761	0.029	4.0	53,497	98.987
G5	0.1	0.093	4.0	315,596	98.437	0.057	4.0	161,301	99.042
	1	0.062	4.1	209,779	98.939	0.040	4.1	100,901	99.323
	10	0.103	4.0	333,123	98.226	0.056	4.0	158,424	99.031
G6	0.1	0.540	4.3	1,980,485	96.885	0.257	4.3	932,838	98.520
	1	0.334	4.6	1,241,202	98.061	0.159	4.6	555,460	99.079
	10	0.518	4.2	1,898,358	97.041	0.247	4.2	877,859	98.590
G7	0.1	0.439	4.0	1,056,146	99.751	0.284	4.0	528,340	99.839
	1	0.343	4.1	851,959	99.793	0.227	4.1	415,395	99.863
	10	0.449	4.0	1,060,658	99.737	0.285	4.0	531,930	99.833
G8	0.1	4.087	4.0	10,558,040	98.864	2.036	4.0	5,238,294	99.434
	1	2.291	4.1	6,492,409	99.344	1.208	4.1	3,196,203	99.654
	10	4.464	4.0	10,679,443	98.761	2.292	4.0	5,366,427	99.364
G9	0.1	62.433	4.7	211,340,810	96.225	26.271	4.7	88,726,983	98.412
	1	43.202	5.7	151,397,990	97.426	15.055	5.7	52,635,184	99.103
	10	71.759	5.4	240,253,623	95.785	27.028	5.4	88,674,631	98.413

the CPU time of the speedup LC algorithm only increases by 2%, 617% and $2,283$%, respectively. Finally, for all the test instances on grid networks, the IPS procedure can find the optimal paths without branching.

5.3. Results for the RC-QDF problem

Finally, we test the PS method for the RC-QDF problem on grid networks. Table 8 gives average computational results of PS1 and PS2 for the RC-QDF problem based on 20 randomly generated instances. The sixth and tenth columns of Table 8 give the improvement in CPU time made by PS1 and PS2 in comparison with the IIL algorithm. Comparison with the OA algorithm is not reported since it does not apply to the RC-QDF problem. From Tables 6 and 8, we observe that the PS method has similar computational performance for both the MSP-SP problem and the RC-QDF problem. Specifically, both PS1 and PS2 reduce the computation time of the IIL algorithm by one to two orders of magnitude, and only a very small number of PSPs need to be solved. In addition, Table 8 shows that the value of θ has an effect on the computation time of the PS

Table 9 Performance of the speedup LC algorithm for the RC-QDF problem on grid networks

		LC		speedup LC		Improvement (%)	
	θ	CPU (ms)	Label	CPU (ms)	Label	CPU	Label
G1	0.1	0.750	2,685.5	0.100	0.3	86.667	99.989
	1	0.550	2,061.1	0.100	2.9	81.818	99.861
	10	0.800	2,608.8	0.050	3.4	93.750	99.870
G2	0.1	1.300	3,735.2	0.100	0.0	92.308	100.000
	1	0.717	2,535.6	0.050	8.5	93.023	99.663
	10	1.100	3,676.9	0.000	0.0	100.000	100.000
G3	0.1	1.500	4,876.4	0.050	5.3	96.667	99.891
	1	1.033	3,071.0	0.050	19.7	95.161	99.359
	10	1.400	5,349.1	0.000	0.0	100.000	100.000
G4	0.1	8.300	26,510.6	0.600	0.0	92.771	100.000
	1	6.233	20,929.5	0.633	5.5	89.840	99.974
	10	8.600	26,665.6	0.450	0.2	94.767	99.999
G5	0.1	22.750	78,874.2	0.350	0.5	98.462	99.999
	1	14.567	51,242.6	0.483	12.3	96.682	99.976
	10	25.550	83,300.6	0.350	10.0	98.630	99.988
G6	0.1	125.617	460,481.8	0.450	138.4	99.642	99.970
	1	71.842	268,495.1	0.617	345.7	99.142	99.871
	10	123.250	450,630.8	0.467	297.1	99.621	99.934
G7	0.1	108.550	263,857.0	3.150	0.1	97.098	100.000
	1	82.367	207,797.2	3.183	3.3	96.135	99.998
	10	113.400	265,345.9	3.150	0.2	97.222	100.000
G8	0.1	1,018.250	2,639,785.3	3.050	0.3	99.700	100.000
	1	556.983	1,582,278.2	3.067	40.2	99.449	99.997
	10	1,113.250	2,670,233.5	2.950	0.4	99.735	100.000
G9	0.1	13,260.808	44,925,091.5	7.258	7,193.1	99.945	99.984
	1	7,622.408	26,667,920.1	13.300	24,871.6	99.826	99.907
	10	13,454.417	44,771,935.8	9.592	11,454.5	99.929	99.974

method. For example, when both attributes have comparable impacts on the objective function, i.e., $\theta = 1$, the PS method has the best computational performance. However, the impact of the value of θ is relatively small compared with that of the network structure, and the PS method still demonstrates a computational advantage over the IIL algorithm for different values of θ and network structures.

Table 9 validates the effectiveness of the speedup LC algorithm for the RC-QDF problem. Similar to the results given in Table 7, the speedup LC algorithm significantly reduces the CPU time and the number of labeling operations of the classical LC algorithm. In addition, Table 9 further shows that when the value of θ deviates from one, the PSPs become more easier to solve for the speedup LC algorithm in terms of both the CPU time and the number of labeling operations. Therefore, the PS method using the speedup LC algorithm is still efficient to solve the RC-QDF problem with unbalanced weights on the attributes. Finally, the IPS procedure also finds the optimal paths for all the test instances of the RC-QDF problem without branching.

6. Conclusions

This paper proposes an effective PS method for the BC-SP problem. The effectiveness of the PS method relies on the number of PSPs required to be solved, and the computation time for these PSPs. To reduce the number of PSPs, we exploit the gradient and concavity of the objective function to speed up the parameter search. A monotonic descent property and an intersection point property have been identified, and MDS and IPS procedures are proposed. Both procedures also apply to similar bicriterion concave minimization problems. To reduce the computation time for PSPs, a speedup LC algorithm is proposed by utilizing optimal paths of previously solved PSPs to reduce the number of labeling operations for the subsequent PSPs. The speedup LC algorithm also serves as an effective subroutine to solve PSPs of other PS methods, such as the gradient based line search for the bicriterion convex minimization problems (Henig 1986, Murthy and Sarkar 1996, Tsaggouris and Zaroliagis 2004). The proposed MDS, IPS and speedup LC algorithm have been further embedded into an interval search algorithm to guarantee global optimality.

Experiments on real transportation networks and grid networks have been conducted to validate the effectiveness of the PS method for the MSD-SP problem and the RC-QDF problem. The proposed method outperforms the most recent exact algorithms, including the IIL algorithm (Khani and Boyles 2015) and the OA algorithm (Shahabi et al. 2013, Shahabi and Boyles 2015). Experimental results also show that only a very small number of PSPs need to be solved, and the computation time for a PSP after the initialization phase is negligible in comparison with that for the first two PSPs. Therefore, the computation time of the PS proposed method can be approximated by $O((2+\epsilon)nm)$, where $0 \leq \epsilon \leq 1$.

Although many shortest path problems in the literature (Wu and Nie 2011) are not special case of the considered BC-SP problem, techniques proposed in this paper may be extended to enhance existing algorithms for these problems. For example, finding an α-reliable shortest path for risk-seeking travelers gives rise to the MSD-SP problem with a negative weight, i.e., $h(x) = \theta\sqrt{x}$ and $\theta < 0$ (Nie and Wu 2009, Chen et al. 2012). Since h is not concave, optimal paths of such problems may not be extreme paths, and thus the PS method can not be directly used to solve this problem. However, when travel times on different links are independent, the problem can be solved by the two-phase methods (Murthy and Sarkar 1996, Tsaggouris and Zaroliagis 2004), which first find the best extreme paths by the PS method and then close the gap using an improved label setting algorithm. Our speedup LC algorithm can be used to reduce the computation time for PSPs in the first phase. Another extension of the proposed PS method is to solve the MSD-SP problem in a stochastic network with limited link travel time correlation. By virtue of the polynomial-time algorithm for the adjacent quadratic shortest path problem proposed by Rostami et al. (2015), the PSPs for this problem can be solved in polynomial time and thus the proposed PS method

applies. For the MSD-SP problem with general link travel time correlation, the OA method can be used to find optimal solutions. Finally, a relevant problem is to find a reliable shortest path in a stochastic time-dependent network. Existing algorithms are based on the first-order stochastic dominance property (Chen et al. 2014). However, it is an open question how to define extreme paths and extend the PS method for this problem. We leave this important extension to future research.

Acknowledgment

The authors thank the editor and anonymous referees for their comments to improve the paper. This work was supported by the National Science Foundation under grant CMMI 1265671, and the National Natural Science Foundation of China under Grants 61503211, 61273233, 71210002 and 71332005, and China Postdoctoral Science Foundation 2015T80102.

References

Bar-Gera, H., 2016. Transportation network test problems. https://github.com/bstabler/TransportationNetworks (Accessed April 2016).

Carstensen, P., 1983. The complexity of some problems in parametric linear and combinatorial programming. PhD thesis, Dept. of Math, University of Michigan.

Chen, B., Lam, W., Sumalee, A., Li, Q., Shao, H., Fang, Z., 2013. Finding reliable shortest paths in road networks under uncertainty. Networks and spatial economics 13 (2), 123–148.

Chen, B., Lam, W., Sumalee, A., Li, Q., Tam, M., 2014. Reliable shortest path problems in stochastic time-dependent networks. Journal of Intelligent Transportation Systems 18 (2), 177–189.

Chen, B., Lam, W., Sumalee, A., Li, Z., 2012. Reliable shortest path finding in stochastic networks with spatial correlated link travel times. International Journal of Geographical Information Science 26 (2), 365–386.

Chen, P., Nie, Y., 2013. Bicriterion shortest path problem with a general nonadditive cost. Transportation Research Part B 57 (5), 419–435.

Climaco, J., Martins, E., 1982. A bicriterion shortest path algorithm. European Journal of Operational Research 11 (4), 399–404.

Climaco, J., Pascoal, M., 2012. Multicriteria path and tree problems: discussion on exact algorithms and applications. International Transactions in Operational Research 19 (1-2), 63–98.

Coutinho-Rodrigues, J., Clímaco, J., Current, J., 1999. An interactive bi-objective shortest path approach: searching for unsupported nondominated solutions. Computers & Operations Research 26 (8), 789–798.

Current, J., Revelle, C., Cohon, J., 1990. An interactive approach to identify the best compromise solution for two objective shortest path problems. Computers & Operations Research 17 (2), 187–198.

Demeyer, S., Goedgebeur, J., Audenaert, P., Pickavet, M., Demeester, P., 2013. Speeding up martins algorithm for multiple objective shortest path problems. 4OR 11 (4), 323–348.

Dial, R., 1979. A model and algorithm for multicriteria route-mode choice. Transportation Research Part B 13 (4), 311–316.

Duque, D., Lozano, L., Medaglia, A., 2015. An exact method for the biobjective shortest path problem for large-scale road networks. European Journal of Operational Research 242 (3), 788–797.

Gabriel, S., Bernstein, D., 1997. The traffic equilibrium problem with nonadditive path costs. Transportation Science 31 (4), 337–348.

Gabriel, S., Bernstein, D., 2000. Nonadditive shortest paths: subproblems in multi-agent competitive network models. Computational & Mathematical Organization Theory 6 (1), 29–45.

Garey, M., Johnson, D., 1979. Computers and intractability: A guide to the theory of np-completeness.

Hansen, P., 1980. Bicriterion path problems. In: Multiple criteria decision making theory and application. Springer, pp. 109–127.

Henig, M., 1986. The shortest path problem with two objective functions. European Journal of Operational Research 25 (2), 281–291.

Horst, R., Pardalos, P., Van Thoai, N., 2000. Introduction to global optimization. Springer Science & Business Media.

Hutson, K., Shier, D., 2009. Extended dominance and a stochastic shortest path problem. Computers & Operations Research 36 (2), 584–596.

Iori, M., Martello, S., Pretolani, D., 2010. An aggregate label setting policy for the multi-objective shortest path problem. European Journal of Operational Research 207 (3), 1489–1496.

Khani, A., Boyles, S., 2015. An exact algorithm for the mean-standard deviation shortest path problem. Transportation Research Part B 81, 252–266.

Lawphongpanich, S., Yin, Y., 2012. Nonlinear pricing on transportation networks. Transportation Research Part C 20 (20), 292–315.

Löfberg, J., 2004. Yalmip: A toolbox for modeling and optimization in matlab. In: Computer Aided Control Systems Design, 2004 IEEE International Symposium on. IEEE, pp. 284–289.

Lozano, L., Medaglia, A., 2013. On an exact method for the constrained shortest path problem. Computers & Operations Research 40 (1), 378–384.

Matthew, C., Kevin, W., 2005. Near-shortest and k-shortest simple paths. Networks 46 (2), 98–109.

Mirchandani, P., Wiecek, M., 1993. Routing with nonlinear multiattribute cost functions. Applied Mathematics and Computation 54 (2-3), 215–239.

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

Murthy, I., Sarkar, S., 1996. A relaxation-based pruning technique for a class of stochastic shortest path problems. Transportation Science 30 (3), 220–236.

Nie, Y., Wu, X., 2009. Shortest path problem considering on-time arrival probability. Transportation Research Part B 43 (6), 597–613.

Nikolova, E., 2009. High-performance heuristics for optimization in stochastic traffic engineering problems. In: International Conference on Large-Scale Scientific Computing. Springer, pp. 352–360.

Nikolova, E., 2010. Approximation algorithms for reliable stochastic combinatorial optimization. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Springer, pp. 338–351.

Nikolova, E., Kelner, J., Brand, M., Mitzenmacher, M., 2006. Stochastic shortest paths via maximization. In: Algorithms-ESA 2006. Springer, pp. 552–563.

Noland, R., Polak, J., 2002. Travel time variability: a review of theoretical and empirical issues. Transport Reviews 22 (22), 39–54.

Paixao, J., Santos, J., 2013. Labeling methods for the general case of the multi-objective shortest path problem - a computational study. In: Computational Intelligence and Decision Making. Springer, pp. 489–502.

Raith, A., 2010. Speed-up of labelling algorithms for biobjective shortest path problems. In: Proceedings of the 45th annual conference of the ORSNZ. Auckland, New Zealand. Proceedings of the 45th annual conference of the ORSNZ. Auckland, New Zealand. pp. 313–322.

Raith, A., Ehrgott, M., 2009. A comparison of solution strategies for biobjective shortest path problems. Computers & Operations Research 36 (4), 1299–1331.

Rostami, B., Malucelli, F., Frey, D., Buchheim, C., 2015. On the quadratic shortest path problem. In: Experimental Algorithms. Springer, pp. 379–390.

Sedeno-Noda, A., Raith, A., 2015. A dijkstra-like method computing all extreme supported non-dominated solutions of the biobjective shortest path problem. Computers & Operations Research 57, 83–94.

Shahabi, M., Boyles, S., 2015. Robust optimization strategy for the shortest path problem under uncertain link travel cost distribution. Computer-Aided Civil and Infrastructure Engineering 30 (6), 433–448.

Shahabi, M., Unnikrishnan, A., Boyles, S., 2013. An outer approximation algorithm for the robust shortest path problem. Transportation Research Part E 58, 52–66.

Sniedovich, M., 1986. C-programming and the minimization of pseudolinear and additive concave functions. Operations Research Letters 5 (4), 185–189.

Thomas, H., Leiserson, C., Rivest, R., Stein, C., 2001. Introduction to algorithms. Vol. 6. MIT press Cambridge.

Tsaggouris, G., Zaroliagis, C., 2004. Non-additive shortest paths. In: European Symposium on Algorithms. Springer, pp. 822–834.

Tuy, H., 1998. Convex analysis and global optimization. Vol. 22. Springer Science & Business Media.

Wu, X., Nie, Y., 2011. Modeling heterogeneous risk-taking behavior in route choice: A stochastic dominance approach. Transportation Research Part A 45 (9), 896–915.

Xie, C., Waller, S., 2012. Parametric search and problem decomposition for approximating pareto-optimal paths. Transportation Research Part B 46 (8), 1043–1067.

Xing, T., Zhou, X., 2011. Finding the most reliable path with and without link travel time correlation: A lagrangian substitution based approach. Transportation Research Part B 45 (10), 1660–1679.

Zeng, W., Miwa, T., Wakita, Y., Morikawa, T., 2015. Application of lagrangian relaxation approach to - reliable path finding in stochastic networks with correlated link travel times. Transportation Research Part C 56, 309–334.

Zhang, C., Chen, X., Sumalee, A., 2011. Robust wardrops user equilibrium assignment under stochastic demand and supply: Expected residual minimization approach. Transportation Research Part B 45 (3), 534–552.

Zhang, Y., Song, S., Shen, Z., 2016a. Data-driven robust shortest path problem with distributional uncertainty. Submitted to IEEE Transactions on Intelligent Transportation Systems.

Zhang, Y., Song, S., Shen, Z., 2016b. Distributionally robust optimization of two-stage lot-sizing problems. Production and Operations Management. Online Available: http://onlinelibrary.wiley.com/doi/10.1111/poms.12602/epdf.