Balancing Profits and Costs on Trees

Sofie Coene
Operations Research Group, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium

Carlo Filippi
Department of Quantitative Methods, University of Brescia, Contrada S. Chiara 50, 25122 Brescia, Italy

Frits C. R. Spieksma
Operations Research Group, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium

Elisa Stevanato
Department of Quantitative Methods, University of Brescia, Contrada S. Chiara 50, 25122 Brescia, Italy

We consider a rooted tree graph with costs associated with the edges and profits associated with the vertices. Every subtree containing the root incurs the sum of the costs of its edges, and collects the sum of the profits of its nodes; the goal is the simultaneous minimization of the total cost and maximization of the total profit. This problem is related to the TSP with profits on graphs with a tree metric. We analyze the problem from a biobjective point of view. We show that finding all extreme supported efficient points can be done in polynomial time. The problem of finding all efficient points, however, is harder; we propose a practical FPTAS for solving this problem. Some special cases are considered where the particular profit/cost structure or graph topology allows the efficient points to be found in polynomial time. Our results can be extended to more general graphs with distance matrices satisfying the Kalmanson conditions. **NETWORKS, Vol. 61(3), 200–211 2013**

Keywords: multiobjective combinatorial optimization; trees; traveling salesman problem with profits; Kalmanson conditions; complexity

1. INTRODUCTION

Here, we deal with a bicriteria combinatorial problem. Given is a rooted tree with costs on the edges and profits on the vertices. Every subtree containing the root incurs a total cost, which is the sum of the costs of its edges, and collects a total profit, which is the sum of the profits of its nodes. We ask for a simultaneous minimization of the total incurred cost and maximization of the total collected profit. Thus, we have two opposite objectives that need to be optimized, one pushing towards not selecting edges to avoid costs, and the other to select edges to collect profit associated with the connected vertices. In this light, our goal is finding a set of Pareto optimal rooted subtrees, i.e., subtrees with the property that neither objective can be improved without deteriorating the other.

The original motivation of this work comes from the study of the traveling salesman problem with profits [14] on a tree metric. Indeed, consider a salesman located at the root of the tree, who wants to visit a subset S of vertices and afterwards returns to the root. He wishes to pay the minimum cost while collecting the maximum profit, and both these goals can be achieved in the following way. For every vertex v in the subset, take the unique path connecting the root to v. Let $T(S)$ be the subtree formed by all such paths. Visit all vertices of $T(S)$ from the root by a complete depth-first search, collecting the profits of each vertex the first time it is encountered. If a vertex in $T(S)$ is not in S, its profit can be collected without affecting the total cost. In the end, every vertex profit in $T(S)$ is collected and every edge is traversed twice, one time while going away from the root, one time during a backtrack. Thus, the route of the salesman incurs a cost which is exactly twice the cost of the corresponding rooted subtree and collects a profit which is exactly the total profit of the subtree. Because there is a one-to-one correspondence between feasible subtours of the salesman and rooted subtrees, preserving the partial ordering with respect to profits and costs, finding Pareto optimal salesman subtours is equivalent to finding Pareto optimal rooted subtrees.

Received December 2008; accepted January 2012
Correspondence to: C. Filippi; e-mail: filippi@eco.unibs.it
Contract grant sponsor: FWO; Contract grant number: G.0541.06
DOI 10.1002/net.21469
Published online 19 May 2012 in Wiley Online Library (wileyonlinelibrary.com).

In general, the TSP with profits is NP-hard [14]. The TSP on a tree, on the other hand, is a straightforward problem to solve (indeed, any tour passes each arc twice). Complexity of the TSP with profits on a tree has, to the best of our knowledge, not been studied before, which makes the problem, as such, interesting to study. In this article, we deal with this complexity issue and separate between easy and hard cases of the TSP with profits on a tree.

Biobjective problems are often dealt with in a single-objective way by either combining the objectives in a single function and/or adding them as a constraint. On one hand, this approach simplifies the analysis but, on the other hand, it hardly captures the complexity of real world decision processes. For instance, Raghavan et al. [33] describe how manufacturing firms are forced to consider multiple criteria when designing products in order to stay competitive. Focusing on a single objective, such as minimizing cost, is no longer sufficient in different markets. Several objectives need to be prioritized and balanced [33]. Accordingly, in the service industry, limiting the objective towards maximizing total profit may not be sufficient in a competitive environment. In the long term, customer service and customer satisfaction yield a more profound basis for a healthy business. That is why it might be interesting to cut down on the number of customers while improving service quality. Thus, a trade-off is made between quantity and quality. That is exactly why a bicriterion approach is interesting and relevant: we make a trade-off between number of customers (i.e., total profit) and service speed (i.e., traveling time).

A first attempt to address the traveling salesman problem with profits as a bicriteria problem was made by Keller and Goodchild [26], who refer to the problem as the multiobjective vending problem; their approach consists of sequentially solving single-objective versions of the problem. Recently, Jozefowiez et al. [22] proposed a metaheuristic method to build up an approximate description of the Pareto optimal solution set and Bérubé et al. [3] developed an exact approach. For an overview on traveling salesman problems with profits, mainly from a single objective point of view, we refer the reader to Feillet et al. [14]. For surveys on multiobjective routing problems we refer to Ehrgott [10] and Jozefowiez et al. [23].

In this work, we consider a biobjective problem where the underlying graph is a tree. It is clear that trees are fundamental network topologies, and many practical problems feature a tree as the underlying graph. More in particular, vehicle routing problems on trees have been discussed in, for instance, Labbé et al. [28], Averbakh and Berman [2], Muslea [30], Lim et al. [29], Chandran and Raghavan [4], and the references contained therein. Motivation for the tree topology comes usually from transportation contexts where the underlying network is a railway network (in pit mines, for instance), a river network, or a sparse road network (in rural areas). Karuno et al. [25] study the problem of scheduling and routing a vehicle, such as an automated guided vehicle, in a building with a simple structure of corridors (each floor corresponds to a subtree and each room to a leaf vertex). In the works mentioned earlier it is required to visit each location. Other work deals with settings where there is a profit to be collected at a location, while not every location needs to be visited. An example is given by Asahiro et al. [1]; they describe a problem where a single server (the vehicle) needs to collect moving items (the locations) on a restricted topology (such as the line). Other examples of such problems are mentioned in Friese and Rambau [15] (elevator scheduling), Psaraftis et al. [32] (scheduling ships along ports on a shoreline), and Coene and Spieksma [6].

In our context, providing the decision maker with the full spectrum of Pareto optimal solutions is not an easy problem, but when we limit the search to extreme supported solutions (a subset of the Pareto optimal solutions), we are able to find them in polynomial time. Further, we show that finding the whole Pareto set is a hard problem, since the problem of finding a single Pareto optimal solution is as hard as the knapsack problem. In a quite general context, approximating the set of Pareto solutions has been dealt with by Papadimitriou and Yannakakis [31]. Their work implies the existence of a so-called FPTAS to find an ϵ-approximate Pareto set for our problem. Constructing such an FPTAS, however, is not straightforward. We exploit the analogy between our problem and a precedence constrained knapsack problem studied by Johnson and Niemi [21] to devise a fast pseudopolynomial dynamic programming algorithm for our biobjective problem. Then, we develop an FPTAS for our problem, using ideas from Erlebach et al. [13] for the multiobjective knapsack problem. Thus, we show that computing the whole Pareto set is more difficult than computing the set of all extreme supported Pareto solutions (assuming $P \neq NP$). Summarizing, our contribution is:

- an $O(n^2)$ algorithm to compute the set of extreme supported Pareto solutions;
- showing that computing the whole Pareto set is hard;
- a pseudopolynomial dynamic programming algorithm to find the Pareto set;
- the explicit description of an FPTAS to find an ϵ-approximate Pareto set;
- an extension of these results to Kalmanson distance matrices.

The article is organized as follows: In Section 2, we define the two problems studied and we give some theoretical background on biobjective optimization. In Section 3, a polynomial time algorithm is developed for finding the set of extreme supported Pareto solutions (referred to as SubtreeESE). Then, in Section 4, we study the complexity of finding all Pareto solutions (SubtreeE) and we develop, as our main contribution, an FPTAS for this problem. Some polynomially solvable special cases are also analyzed. In Section 5, we show how our results are extendable to graphs with a Kalmanson distance matrix. Section 6 provides concluding remarks.

2. PROBLEM DEFINITION

In this section, we define the biobjective optimization problem we are interested in. We refer to Ehrgott [11] and

T'kindt and Billaut [34] for an introduction into multicriteria optimization.

Let $T = (V, E)$ be a tree where $V = \{0, 1, 2, \ldots, n\}$ is a set of $n + 1$ vertices and E is a set of edges. We consider vertex 0 as the root. Let a profit p_i be associated with each vertex $i \in V$ and a cost c_{ij} be associated with each edge $(i, j) \in E$. We will assume that profits and costs are strictly positive, with the only exception that $p_0 = 0$.

A feasible subtree $S = (V(S), E(S))$ is a subtree of T containing the root. The profit of a feasible subtree is:

$$p(S) = \sum_{i \in V(S)} p_i,$$

whereas the cost of S equals

$$c(S) = \sum_{(i,j) \in E(S)} c_{ij}.$$

If $S = (\{0\}, \emptyset)$, the feasible subtree corresponds to the solution with zero cost and zero profit. Note that we assume a given root, but all positive results we state also hold for the case where one is allowed to choose the root position (at the expense of a factor n in the running times).

A feasible subtree S is Pareto optimal if there exists no feasible subtree S' such that $c(S') \leq c(S)$ and $p(S') \geq p(S)$, and at least one inequality is strict. A point $(\gamma, \pi) \in \mathbb{R}^2$ is an efficient point if there exists a Pareto optimal subtree S such that $c(S) = \gamma$ and $p(S) = \pi$. Let $\mathcal{E}$ denote the set of efficient points. In Figure 1a every point represents a feasible solution associated with a subtree S. Cost of a subtree is represented on the horizontal axis, profit on the vertical axis. The filled circles correspond to efficient points.

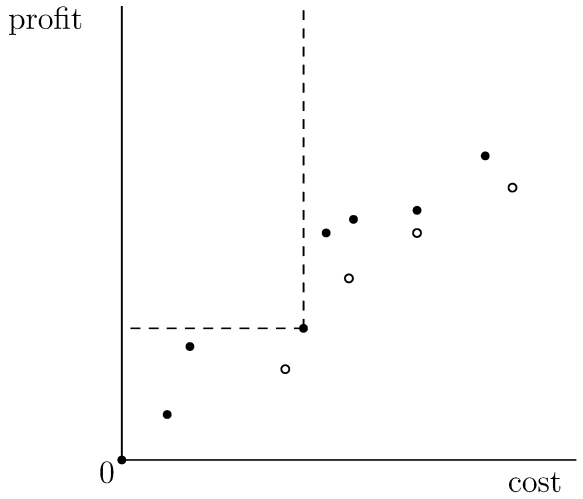
An efficient point $(\gamma, \pi) \in \mathbb{R}^2$ is supported if there exists a scalar $\lambda \in [0, 1]$ and a feasible subtree S such that S maximizes $\lambda p(S) - (1 - \lambda)c(S)$ and $c(S) = \gamma$, $p(S) = \pi$. A supported efficient point that is an extreme point of the convex hull of all solution points, is called an extreme supported efficient point (Hansen [19], Ehrgott [11]). Let $\mathcal{SE}$ denote the set of extreme supported efficient points (see Fig. 1b). Clearly, $\mathcal{SE} \subseteq \mathcal{E}$.

The goal of our work is to investigate the difficulty of this bicriteria subtree problem. We distinguish the following two tasks.

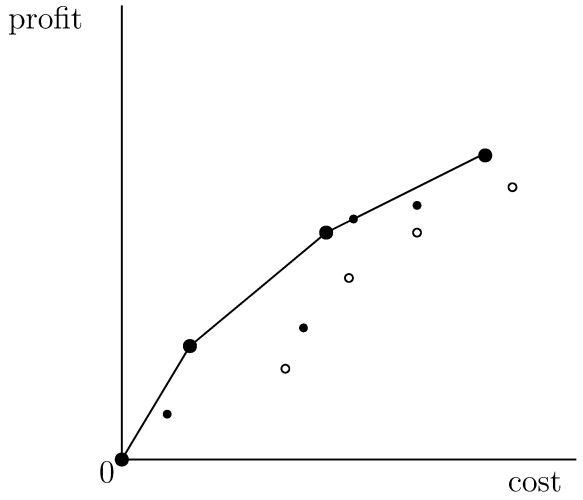
SubtreeE ($\mathcal{E}$). Find all efficient points and, for each of them, a corresponding Pareto optimal subtree.

SubtreeESE ($\mathcal{SE}$). Find all extreme supported efficient points and, for each of them, a corresponding Pareto optimal subtree.

Although the topology of our setting is restricted, the questions are ambitious because —in both problems— we are not aiming for a single solution, but instead for a set of solutions. Of course, to do so in polynomial time, the number of points in the set should be polynomial in the input size; we come back to this issue in Section 4.1. Note that the above problems are sorted by decreasing complexity. Indeed, computing the set of extreme supported efficient points is not harder than computing the set of efficient points, because the latter set of points contains the former. We prove that, assuming that $P \neq NP$, SubtreeE is more difficult than SubtreeESE. Similar results for the biobjective shortest path problem were obtained by Henig [20]. We also show that our results still hold in a more general setting where the tree metric is generalized to distance matrices satisfying the Kalmanson conditions.



(a) Filled circles are efficient points



(b) Bold filled circles are extreme supported efficient points

FIG. 1. Efficient points (a) and extreme supported efficient points (b).

The problems introduced above may be related to biobjective minimum spanning tree (MST) problems [7]. There is, however, a crucial difference: in biobjective MST problems we have to span all vertices of a given general graph, whereas we are interested in subtrees of a given tree.

3. PROBLEM SUBTREEESE

In this section, we show that finding all extreme supported efficient points, i.e., solving SubtreeESE, can be done

in $O(n^2)$ time by using a parametric linear program with a very special structure. The structure is such that an algorithmic idea developed by Eisner and Severance [12] can be used to solve our problem efficiently. Several methods for solving combinatorial parametric programming problems are discussed in Gusfield [18].

Let $T = (V, E)$ be a tree, rooted at vertex 0. For all $i \in V \setminus \{0\}$, let $i^\star$ denote the parent of i along the unique path from i to 0. Further, we denote by $\delta(i) \subseteq V$ the set of children of i, $i \in V$; if i is a leaf then clearly $\delta(i) = \emptyset$. We assume that a summation over an empty set equals 0. We associate with every vertex i a binary variable x_i, which equals 1 if and only if i belongs to a feasible subtree. Notice that $x_i = 1$ implies $x_{i^\star} = 1$. SubtreeESE corresponds to describing the solutions of the following parametric program, where $\lambda \in [0, 1]$ is the parameter:

$$\begin{array}{rll} \max & \sum_{i \in V \setminus \{0\}} (\lambda p_i - (1-\lambda) c_{ii^\star}) x_i & \\ \text{subject to} & x_0 = 1 & \\ & x_i - x_{i^\star} \leq 0 & (i \in V \setminus \{0\}) \\ & x_i \in \{0, 1\} & (i \in V) \end{array} \tag{1}$$

Vertices are selected such that the sum of the weighted differences between the profits and costs is maximized. A feasible solution consists in a set of connected vertices which are also connected to the root. This is enforced in the constraints in (1). The coefficient matrix of problem (1) is a totally unimodular matrix, because it is the transpose of the node-arc incidence matrix of tree T, where all edges are oriented to the root. Thus we may relax the integrality constraints to non-negativity constraints. The dual of the relaxed problem is the following:

$$\begin{array}{rll} \min & y_0 & \\ \text{subject to} & y_0 \geq \sum_{j \in \delta(0)} y_j & \\ & y_i \geq (\lambda p_i - (1-\lambda) c_{ii^\star}) & \\ & \quad + \sum_{j \in \delta(i)} y_j & (i \in V \setminus \{0\}) \\ & y_i \geq 0 & (i \in V \setminus \{0\}) \end{array} \tag{2}$$

An optimal solution of problem (2) can be described as a function of λ:

$$y_i(\lambda) = \max\{0; (\lambda p_i - (1-\lambda) c_{ii^\star}) + \sum_{j \in \delta(i)} y_j(\lambda)\} \quad (i \in V \setminus \{0\}) \tag{3}$$

In particular, $y_0(\lambda)$ is the value function of problem (2), i.e., the function describing the optimal value of our parametric program.

Eisner and Severance [12] suggest a simple and elegant algorithm for parametric problems with a piecewise linear and convex value function having a finite number of breakpoints. This algorithm needs to evaluate the value function at most $O(k)$ times, with k the number of breakpoints. Thus, if the value function can be evaluated in polynomial time for any fixed parameter value, then it can be described in time polynomial in the number of breakpoints. We show that Eisner and Severance's algorithm can be applied to solve efficiently SubtreeESE.

First, we note that for any fixed value of λ, the unique solution of equations (3), and in particular $y_0(\lambda)$, can be computed in $O(n)$ time by going backwards from the leaves to the root. Furthermore, by expanding equation (3) recursively, one can see that $y_i(\lambda)$ is a nondecreasing, piecewise linear and convex function of the parameter λ, for all $i \in V$.

Second, we argue that the number of breakpoints of $y_0(\lambda)$ is at most $n - 1$. For all λ for which $y_0(\lambda) = 0$, the optimal subtree contains only vertex 0. Otherwise, by complementary slackness, an optimal subtree visits all vertices i that have $y_i(\lambda) > 0$ as well as $y_j(\lambda) > 0$ for each j along the unique path from i to 0. Alternatively, consider the set-valued function $E^\star : [0, 1] \mapsto 2^E$, where $E^\star(\lambda) = \{(i, i^\star) \in E : y_i(\lambda) > 0\}$, and let $T(\lambda) = (V, E^\star(\lambda))$. For any given λ, an optimal subtree visits the vertices belonging to the connected component of $T(\lambda)$ containing the root (vertex 0). Thus, to describe function $y_0(\lambda)$ and to enumerate all extreme supported subtours, it is sufficient to record the different sets of function $E^\star$.

Suppose that we increase parameter λ continuously from 0 to 1, and let $\lambda' < \lambda''$. Because $y_i(\lambda)$ is nondecreasing for all i, if $(i, i^\star) \in E^\star(\lambda')$ then $0 < y_i(\lambda') \leq y_i(\lambda'')$, and hence $(i, i^\star)$ remains in $E^\star(\lambda'')$. Thus, $E^\star(\lambda') \subseteq E^\star(\lambda'')$. Furthermore, since $E^\star(0) = \emptyset$ (corresponding to the trivial subtree consisting of vertex 0) and $E^\star(1) = E$ (corresponding to the complete tree with all vertices), $E^\star$ changes its value in only $k \leq n - 1$ breakpoints, $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_k < 1$. It follows that the number of extreme supported efficient points is constrained by n.

Theorem 1. *SubtreeESE is solvable in $O(n^2)$ time.*

Proof. We have shown that SubtreeESE can be solved using a parametric program with a piecewise linear convex function and a finite number of breakpoints. In such a setting the algorithm from [12] can be applied. Evaluating $y_0(\lambda)$ for any fixed λ takes $O(n)$ time and this evaluation needs to take place $O(n)$ times, yielding a total time complexity of at most $O(n^2)$. ∎

Notice also that we can solve SubtreeESE on the line in $O(n)$. On a line it holds that when a vertex i belongs to a Pareto optimal subtree, all vertices positioned between the root 0 and i belong to that subtree as well. Further, no other nonchild vertex of i will be served without having to pass through the root. Notice that on a general tree this does not hold; another leaf vertex could be served in the same subtree. It follows that, using (3), we are able to compute a critical value

$$\lambda_i = \frac{c_{ii^\star}}{p_i + c_{ii^\star}}$$

for each vertex i, such that when $\lambda > \lambda_i$ we have $y_j(\lambda) > 0$ for all j on the $0 - i$ path. Consider the vertices $1, \ldots, n_L$ on

the left side of the root and the corresponding critical values. When $\lambda_i > \lambda_{i+1}$, the $0-i$ path will never be a Pareto optimal subtree. By going through the critical values from λ_{n_L} to λ_1, all $\lambda_i \geq \lambda_j$ with $i < j \leq n_L$ can be removed. The same process can be repeated for the vertices on the other side of the root. The remaining critical values are sorted in increasing order and give the Pareto optimal subtrees.

4. PROBLEM SUBTREEE

This section deals with SubtreeE, the problem of computing all efficient points. Subsection 4.1, is devoted to the complexity of this problem. In Subsection 4.2, we describe a (pseudopolynomial) dynamic program that computes the efficient points, while Subsection 4.3 proposes an FPTAS. Subsection 4.4 is devoted to some polynomially solvable special cases.

4.1. Complexity

Traditional complexity theory deals with problems where the requested output (usually a single solution) is polynomial in the given input. However, when the goal is to find a set of solutions instead of a single one, it may well be the case that the requested output is exponential in the given input. The computational complexity of bicriteria problems has been addressed in Garey and Johnson [17], T'Kindt et al. [35], Fukuda [16] and the references contained therein. Here, we will argue that it is unlikely that there is a polynomial algorithm for finding all efficient points of SubtreeE. We do this by relating the problem to the well-known knapsack problem; recall that the knapsack problem is defined by a set of n items, each item having weight w_i and a value q_i and a knapsack with capacity B, and the problem consists in finding a maximum-valued subset of the items whose total weight does not exceed the capacity B. Let us refer to the problem of finding an optimal solution for each value of B, as problem all-KnapSack.

Theorem 2. *SubtreeE is at least as hard as all-KnapSack*

Proof. Let us build an instance of SubtreeE by constructing a tree consisting of $n+1$ vertices and n edges such that each edge connects the root (vertex 0) with a vertex i, $1 \leq i \leq n$; the resulting graph is known as a star. Assign to each edge $(i,0)$ a cost $c_{i0} = w_i$, associate with each vertex $i \neq 0$ a profit $p_i = q_i$. Observe now that a feasible knapsack solution that is an optimal solution for some value of the capacity B, is a Pareto optimal solution for SubtreeE, and vice versa. ∎

4.2. A Dynamic Programming Algorithm for SubtreeE

In this section, we review the "left–right" dynamic programming algorithm by Johnson and Niemi [21] for the out-tree knapsack problem, revised to fit SubtreeE.

Let $0, 1, 2, \ldots, n$ be a depth-first ordering of the vertices of tree $T = (V, E)$, starting with the root. Let $d(i) = |\delta(i)|$ be the number of children of node i, going away from the root. Obviously, if the node i is a leaf then $d(i) = 0$.

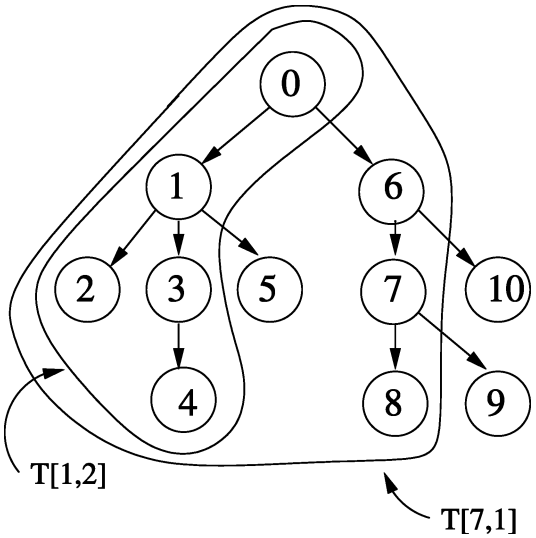


FIG. 2. Examples of subtree in left–right approach.

For each $i \in V$ and $0 \leq s \leq d(i)$, we define $T[i,s]$ as the subtree of T induced by i, the first s children of i taken in order of index, all their successors, and all vertices in V with index lower than i (see Fig. 2).

We define the left-right ordering of the subtrees as follows:

a. $T[i,s]$ precedes $T[i,s+1]$ for all $i \in V$ and $s \in \{1, \ldots, d(i)-1\}$;
b. if j is the sth child of i then $T[i,s-1]$ precedes $T[j,0]$ and $T[j,d(j)]$ precedes $T[i,s]$ in the ordering.

Note that with the above ordering, every subtree contains all the subtrees that precede it. From the initial tree $T[0,0] = (\{0\}, \emptyset)$ we gradually expand first down the left edge of the tree and then across the tree to the right. Notice further that some trees may be identical. More precisely, if j is the sth child of i then $T[j,d(j)] = T[i,s]$. In other words, each time we have to backtrack while searching the tree in depth-first order, we replicate the same subtree. Anyway, we consider all defined subtrees as distinct objects. As a consequence, the total number of considered subtrees is exactly $2n+1$.

A q-subtree for $T[i,s]$ is a feasible subtree (i.e., a subtree containing the root) that visits vertex i, cannot visit vertices out of $T[i,s]$ and has total profit equal to q. Note that, in the previous example, although $T[j,d(j)] = T[i,s]$, a q-subtree for $T[j,d(j)]$ must contain j, while a q-subtree for $T[i,s]$ needs not.

Let $C_{\text{tot}} = \sum_{(i,j)\in E} c_{ij}$ and $P_{\text{tot}} = \sum_{i\in V} p_i$. For each triple $[i,s,q]$ with $i \in \{0,1,\ldots,n\}$ and $s \in \{0,1,\ldots,d(i)\}$ we define $C[i,s,q]$ as the minimum cost of a q-subtree for $T[i,s]$. If there is no q-subtree for $T[i,s]$ then we define $C[i,s,q] = \infty$. This obviously happens for $q < 0$ and $q > P_{\text{tot}}$.

We compute function C by the following algorithm, which is a restatement of the left–right approach in [21].

Algorithm LR-DP

1. (Initialization) $C[0,0,0] = 0$; $C[0,0,q] = \infty$ for $q = 1,2,\ldots,P_{\text{tot}}$;
2. (Recursion) for all subtrees $T[i,s]$ sorted in left-right order:
   for all $q = 0,1,\ldots,P_{\text{tot}}$:
   (a) if $s = 0$ then

$$C[i,0,q+p_i] = C[k,z-1,q] + c_{k,i} \qquad (4)$$

   where i is the zth child of k;
   (b) if $s \in \{1,\ldots,d(i)\}$ then

$$C[i,s,q+p_i] = \min\{C[i,s-1,q+p_i], C[j,d(j),q+p_i]\}; \qquad (5)$$

   where j is the sth child of i.

The recursion step can be explained as follows. In step 2(a), i is the largest element in the subtree and, by definition, i will be visited. Thus, the cost of this state is equal to the cost of a feasible subtree with profit q in the largest subtree of T not containing i, increased with the extra cost of visiting i. In step 2(b), either the feasible subtree with profit $q + p_i$ visits the sth child of i, vertex j, or it does not. If j is not visited, then we have to look for a minimum cost $(q + p_i)$-subtree for $T[i, s-1]$, i.e., the largest subtree in left–right ordering not containing j. If vertex j is visited, then we have to look for a minimum cost $(q + p_i)$-subtree for $T[j, d(j)]$, that by definition contains j.

The advantage of algorithm LR-DP with respect to more intuitive dynamic programming procedures, based on a "bottom-up" search strategy of the tree, is that the evaluation of every entry of the array C takes a constant time. As a consequence, the complexity of LR-DP is linear in the number of entries in C, which is $(2n+1)P_{\text{tot}}$.

It is easy to verify that the values $C[i,s,q]$ returned by algorithm LR-DP satisfy the definition. As a consequence, the collection of ordered pairs

$$(C[0,d(0),0],0),(C[0,d(0),1],1),\ldots(C[0,d(0),P_{\text{tot}}],P_{\text{tot}}),$$

contains the efficient set $\mathcal{E}$. We can filter the efficient points from this ordered list in linear time as follows.

Go through the list in decreasing order of the second coordinate q. Consider two consecutive points $C[0,d(0),q']$ and $C[0,d(0),q'']$, with $q' > q''$. If $C[0,d(0),q''] \geq C[0,d(0),q']$ then eliminate $C[0,d(0),q'']$ and repeat with $q'' = q'' - 1$. Otherwise, repeat with $q' = q''$ and $q'' = q'' - 1$.

To reconstruct the Pareto optimal subtrees associated with the efficient points just computed, we keep track of the computations done in algorithm LR-DP by using a new function W with the same domain as C. More precisely, in step 2(a) of algorithm LR-DP, after computing the value of $C[i,0,q+p_i]$, we set $W[i,0,q+p_i] = (k,z-1,q)$, where i is the zth child of k; in step 2(b), after computing the value of $C[i,s,q+p_i]$, if $C[i,s,q+p_i] = C[i,s-1,q+p_i]$ then we set $W[i,s,q+p_i] = (i,s-1,q+p_i)$, if $C[i,s,q+p_i] = C[j,d(j),q+p_i]$, where j is the sth child of i, then we set $W[i,s,q+p_i] = (j,d(j),q+p_i)$. For example, as the value of $C[1,0,p_1]$ derives from $C[0,0,0]$, we set $W[1,0,p_1] = (0,0,0)$.

At the end of algorithm LR-DP, starting from an entry of W corresponding to an efficient point (γ,π), we initialize a vertex subset $V' = \{0\}$. Then, we backtrack on the entries of W until we reach the entry equal to $(0,0,0)$. At each step, if we find an entry $(i,0,q)$, then we add i to V'; if we find an entry $(j,d(j),q)$, then we add j to V'. When we reach $(0,0,0)$, V' contains the vertices that form the Pareto optimal subtree corresponding to (γ,π).

This procedure takes at most $O(n)$ steps for every efficient point; this implies that, in the worst case, we can compute all Pareto optimal subtrees in $O(nP_{\text{tot}})$.

We summarize the above discussion by the following statement.

Theorem 3. *SubtreeE can be solved in $O(nP_{\text{tot}})$ time.*

4.3. An FPTAS for SubtreeE

It is well-known that the existence of pseudopolynomial dynamic programs lead, under certain conditions, to the existence of polynomial time approximation schemes, see Woeginger [36]. From Papadimitriou and Yannakakis [31] we know that such a polynomial time approximation scheme must exist for our bicriteria setting. However, the results presented in [31] are based on general search strategies of the criteria space and do not directly translate to practical algorithms.

In this section, we investigate how to approximate the efficient set in an effective way. We start with a few definitions.

For $\epsilon \geq 0$, a pair (γ,π) is called an ϵ-approximation of a pair $(\gamma^\star,\pi^\star)$ if $\gamma \leq (1+\epsilon)\gamma^\star$ and $\pi \geq \pi^\star/(1+\epsilon)$. A set $\mathcal{E}'$ of points in the cost-profit space is called an ϵ-approximation of the efficient set $\mathcal{E}$ if, for every $(\gamma^\star,\pi^\star) \in \mathcal{E}$ there exists a point $(\gamma,\pi) \in \mathcal{E}'$ such that (γ,π) is an ϵ-approximation of $(\gamma^\star,\pi^\star)$. Notice that the closer ϵ is to zero, the better the approximation of the efficient set. See Papadimitriou and Yannakakis [31] for more information on approximating multicriteria problems.

An algorithm that runs in polynomial time in the size of the input and that always outputs an ϵ-approximation of the efficient set is called an ϵ-approximation algorithm. A polynomial time approximation scheme (PTAS) for the efficient set is a family of algorithms that contains, for every fixed constant $\epsilon > 0$, an ϵ-approximation algorithm A_ϵ. If the running time of A_ϵ is polynomial in the size of the input and in $1/\epsilon$, the family of algorithms is called a fully polynomial time approximation scheme (FPTAS).

In this section, we develop an FPTAS for SubtreeE. We use the standard idea for developing an FPTAS for a knapsack problem, i.e., we scale the profits and apply an exact dynamic programming approach with the scaled profits. An FPTAS for the out-tree knapsack based on this idea is suggested in

[21]. However, such a scheme does not translate directly to an FPTAS for SubtreeE. Indeed, in [21] a classical partitioning of the profit space into intervals of equal size is used, that guarantees a bound on the absolute error on every generated point. The bound is chosen so that when the maximum admissible cost (weight) is reached, the relative error ϵ is guaranteed. However, in order to get an ϵ-approximation of the efficient set, we require an algorithm that computes a feasible solution with a relative error ϵ for every possible cost and profit value. To this end, we use the partition of the profit space suggested by Erlebach et al. [13] for the multiobjective knapsack problem.

We partition the profit space in u intervals:

$$[1,(1+\epsilon)^{1/n}),[(1+\epsilon)^{1/n},(1+\epsilon)^{2/n}),$$
$$[(1+\epsilon)^{2/n},(1+\epsilon)^{3/n}),\ldots,[(1+\epsilon)^{(u-1)/n},(1+\epsilon)^{u/n})$$

with $u=\lceil n\log_{1+\epsilon}P_{\text{tot}}\rceil$. Notice that the union of all intervals generates the whole profit range $[1,P_{\text{tot}})$, and that u is of order $O(n(1/\epsilon)\log P_{\text{tot}})$[1], hence polynomial in the length of the input and in $1/\epsilon$. We can see that in every interval, the upper endpoint is $(1+\epsilon)^{1/n}$ times the lower endpoint. Then, we adapt algorithm LR-DP to the new interval profit space. We consider as profits the value 0 and the u lower endpoints of the intervals. For convenience, we denote by ℓ_w the lower endpoints, with $w=1,2,\ldots,u$, and we define $\ell_0=0$.

A scaled q-subtree for $T[i,s]$ is a feasible subtree that visits vertex i, cannot visit vertices out of $T[i,s]$ and has total profit q or more.

Instead of C, we consider a different function, denoted $\widetilde{C}$. For each triple $[i,s,\ell_w]$ with $i\in\{0,1,\ldots,n\}$, $s\in\{0,1,\ldots,d(i)\}$, and $w\in\{0,1,\ldots,u\}$, we define $\widetilde{C}[i,s,\ell_w]$ as the minimum cost of a scaled ℓ_w-subtree for $T[i,s]$. If there is no scaled ℓ_w-subtree for $T[i,s]$, then we define $\widetilde{C}[i,s,\ell_w]=\infty$.

Algorithm Scaled-LR-DP

1. (Initialization) $\widetilde{C}[0,0,\ell_0]=0$; $\widetilde{C}[0,0,\ell_w]=\infty$ for $w=1,2,\ldots,u$;
2. (Recursion) for all subtrees $T[i,s]$ sorted in left–right order:
   for all $w=0,1,\ldots,u$:
   let $r=\max\{j:\ell_j\le\ell_w+p_i\}$ (i.e., ℓ_r is the largest lower endpoint not greater than ℓ_w+p_i);
   (a) if $s=0$ then
   $$\widetilde{C}[i,0,\ell_r]=\widetilde{C}[k,z-1,\ell_w]+c_{k,i} \qquad (6)$$
   where i is the zth child of k;
   (b) if $s\in\{1,\ldots,d(i)\}$ then
   $$\widetilde{C}[i,s,\ell_r]=\min\{\widetilde{C}[i,s-1,\ell_r],\widetilde{C}[j,d(j),\ell_r]\} \qquad (7)$$
   where j is the sth child of i.
3. (Output) return the points
   $$(\gamma_w,\pi_w)=(\widetilde{C}[0,d(0),\ell_w],\ell_w) \qquad (w=0,1,\ldots,u).$$

To obtain the feasible subtrees corresponding to the returned points in the cost-profit space, we may use the array W already described for algorithm LR-DP. Notice that in Scaled-LR-DP we directly return the last row of the array $\widetilde{C}$, containing only efficient points (in every state we calculate the cost when profit is equal or larger than ℓ_w).

Theorem 4. *Algorithm Scaled-LR-DP is an FPTAS for SubtreeE with time complexity $O(n^2(1/\epsilon)\log(P_{\text{tot}}))$.*

Proof. This proof follows the lines used in Erlebach et al. [13].

Let the subtrees be numbered according to the left–right ordering, i.e.,

$$T_0=T[0,0],\ T_1=T[1,0],\ldots,\ T_{2n-1}=T[0,d(0)].$$

We show that algorithm Scaled-LR-DP returns an ϵ-approximation of the efficient set. More precisely, we show the following claim.

Claim 1. *For every $m\in\{1,2,\ldots,2n+1\}$, let $T_m=T[i,s]$, and let h be the largest index of a vertex belonging to T_m. After performing the Recursion step on $T[i,s]$, for the optimal cost function C and the approximate cost function $\widetilde{C}$ there exists, for every entry $C[i,s,q+p_i]$ an entry $\widetilde{C}[i,s,\ell_r]$ such that:*

a. $\widetilde{C}[i,s,\ell_r]\le C[i,s,q+p_i]$,
b. $(1+\epsilon)^{h/n}\ell_r\ge q+p_i$.

Notice that when $m=2n+1$ then $T[i,s]=T[0,d(0)]$, $h=n$, and conditions (a) and (b) above imply that the point set returned by algorithm Scaled-LR-DP is an ϵ-approximation of the efficient set.

We prove the claim by induction on the tree index m.

The basis of the induction is given by $T_1=T[1,0]$, where $h=1$. We distinguish between two cases:

CASE $q=0$. We have:

$$C[1,0,p_1]=C[0,0,0]+c_{0,1}=c_{0,1}=\widetilde{C}[0,0,0]+c_{0,1}$$
$$=\widetilde{C}[1,0,\ell_r]$$

where $\ell_r=\max\{j:\ell_j\le p_1\}$. This proves that property (a) holds with equality.

Property (b) follows from the fact that p_1 and ℓ_r are in the same interval, hence:

$$(1+\epsilon)^{1/n}\ell_r\ge p_1$$

CASE $q\ge 1$. We have:

$$C[1,0,q+p_1]=C[0,0,q]+c_{0,1}=\infty.$$

[1] $\log_a b=\frac{\log b}{\log a}$; $\lim_{x\to 0}\log(1+x)=x$

Let $r = \max\{j : \ell_j \le q + p_1\}$ and $w = \max\{1, \min\{j : \ell_r \le \ell_j + p_1\}\}$. Then, $r = \max\{j : \ell_j \le \ell_w + p_1\}$ as required by step 2 of algorithm Scaled-LR-DP, and $\ell_w \ge 1$, so that

$$\widetilde{C}[1, 0, \ell_r] = \widetilde{C}[0, 0, \ell_w] + c_{0,1} = \infty.$$

Then property (a) holds. Furthermore, ℓ_r and $q + p_i$ are in the same interval, so that:

$$(1 + \epsilon)^{1/n} \ell_r \ge q + p_1$$

This ends the basis of the induction.

Assume that the claim is true for any m, $1 \le m < 2n + 1$ and consider $T_{m+1} = T[i, s]$. Again, we distinguish between two cases.

CASE $s = 0$. In this case the highest index in $T_{m+1} = T[i, 0]$ is given by vertex i, then $h = i$. Property (a) follows from (4) and (6):

$$\begin{aligned} C[i, 0, q + p_i] &= C[k, z - 1, q] + c_{k,i} \\ &\ge \widetilde{C}[k, z - 1, \ell_w] + c_{k,i} = \widetilde{C}[i, 0, \ell_r] \end{aligned}$$

where the inequality holds by the inductive hypothesis and $r = \max\{j : \ell_j \le \ell_w + p_1\}$. Now, we consider property (b). By the induction hypothesis, the claim holds with $T[k, z-1]$, where $h = i - 1$, then:

$$(1 + \epsilon)^{(i-1)/n} \ell_w \ge q.$$

Hence, we have:

$$\ell_w + p_i \ge q/(1 + \epsilon)^{(i-1)/n} + p_i \ge (q + p_i)/(1 + \epsilon)^{(i-1)/n}$$

and as ℓ_r and $\ell_w + p_i$ are in the same interval, with ℓ_r as lower bound, it holds that

$$(1 + \epsilon)^{1/n} \ell_r \ge \ell_w + p_i \ge (q + p_i)/(1 + \epsilon)^{(i-1)/n}.$$

From these relations property (b) follows immediately:

$$(1 + \epsilon)^{i/n} \ell_r \ge q + p_i.$$

CASE $s > 0$. From (5) we have $C[i, s, q + p_i] = C[i, s - 1, q + p_i]$ or $C[i, s, q + p_i] = C[j, d(j), q + p_i]$.

In the first case, we have

$$\begin{aligned} C[i, s, q + p_i] = C[i, s - 1, q + p_i] &\ge \widetilde{C}[i, s - 1, \ell_r] \\ &\ge \widetilde{C}[i, s, \ell_r] \end{aligned}$$

for some lower endpoint ℓ_r, where the first inequality follows from the induction hypothesis and the second inequality follows from (7). This proves property (a). Concerning property (b), let k be the largest index of a vertex in $T[i, s-1]$. Clearly, $h > k$, so that

$$(1 + \epsilon)^{h/n} \ell_r \ge (1 + \epsilon)^{k/n} \ell_r \ge q + p_i$$

where the second inequality follows from the inductive hypothesis.

In the second case, we have similarly

$$\begin{aligned} C[i, s, q + p_i] = C[j, d(j), q + p_i] &\ge \widetilde{C}[j, d(j), \ell_r] \\ &\ge \widetilde{C}[i, s, \ell_r] \end{aligned}$$

for some lower endpoint ℓ_r, where the first inequality follows from the induction hypothesis and the second inequality follows from (7). This again proves property (a). As already noticed, $T[i, s]$ and $T[j, d(j)]$ are in fact the same tree, so that the largest index h of a vertex in both trees is the same. Hence, property (b) follows directly from the inductive hypothesis.

From Theorem 3 and the bound on the number of different lower endpoints it follows that the complexity of Scaled-LR-DP is $O(n^2(1/\epsilon)\log(P_{\text{tot}}))$. ∎

4.4. Some Special Cases

In this section we analyze some special cases of trees or cost/profit structures where SubtreeE is polynomially solvable. We deal with equal profits and/or equal costs (Subsection 4.4.1) and the line (Subsection 4.4.2).

4.4.1. Trees with Equal Profits or Equal Costs. We have proven that SubtreeE is hard, even on a star. However, Theorem 3 implies that in the special case where the sum of all the profits P_{tot} is polynomial in n, SubtreeE is polynomially solvable.

If all vertex profits are equal (but the edge costs are arbitrary), we may assume $p_i = 1$ for all $i \in V \setminus \{0\}$, so that $P_{\text{tot}} = n$. In this case, algorithm LR-DP solves SubtreeE in $O(n^2)$ time.

If all edge costs are equal (but the vertex profits are arbitrary), we may assume $c_{ij} = 1$ for all $(i, j) \in E$. In this case, the cost of a feasible subtree is the number of visited vertices. Hence there are at most $n + 1$ efficient points (exactly $n + 1$ since we assumed that all vertex profits are strictly positive). Then we can exchange the role of profits and costs as follows. For all $i \in V \setminus \{0\}$, let $i^\star$ denote the parent of i along the unique path from i to 0. Let $M = \max_i\{p_i\} + 1$. We modify the edge costs and the vertex profits as follows:

$$\begin{aligned} c'_{ii^\star} &= -p_i + M \quad \text{for all } i \in V \setminus \{0\}, \\ p'_i &= 1 \qquad\quad\ \ \text{for all } i \in V \setminus \{0\}. \end{aligned}$$

With the modified costs and profits we are back to the case of general costs and equal profits considered above. It is easy to see that there is a one-to-one correspondence between the efficient points (and corresponding feasible subtrees) in the modified problem and the original one. Thus SubtreeE in the case of equal costs and arbitrary profits on a tree can still be solved in $O(n^2)$ time by algorithm LR-DP.

4.4.2. The Line. Let $T = (V, E)$ be a line, let the edge costs be arbitrary positive and let the vertex profits be arbitrary positive (except $p_0 = 0$). We distinguish between two cases.

If the root is an extreme point, we may assume that the vertices are numbered from left to right, so that the root 0 is the

leftmost vertex. In this case, it is easy to see that the n feasible subtrees $(0,1), (1,2), (i-1,i)$ for $i = 0,\ldots,n$ are all Pareto optimal solutions. Here, a feasible subtree is identified by its rightmost vertex i. Thus, in this case SubtreeE is solvable in $O(n)$ time.

If, however the root is not an extreme point then SubtreeE is a little less straightforward to solve. We start by defining a lower bound on the number of efficient points.

Theorem 5. *SubtreeE on the line can have $O(n^2)$ efficient points.*

Proof. Consider the following instance of SubtreeE on the line. We are given a set of n vertices and a single root positioned on the line. Exactly $\lfloor n/2 \rfloor$ vertices are positioned to the left of the root (called the left vertices), and $\lceil n/2 \rceil$ vertices are positioned to the right of the root (called the right vertices). For each left vertex i, we have $p_i = 1$, and each distance between a pair of consecutive left vertices i and $i+1$ equals $c_{i,i+1} = 1$. For each right vertex i, we have $p_i = n$, and each distance between a pair of consecutive right vertices i and $i+1$ equals $c_{i,i+1} = n$. It follows that for every pair consisting of a left vertex i and a right vertex j, the feasible subtree that ranges from i to j is a Pareto optimal solution. ∎

An algorithm for this case is the following. For each pair of left and right vertices, compute profit and cost of the corresponding subtree. Sort the obtained points, e.g., with respect to profit, and then filter the efficient points. This takes $O(n^2 \log n)$ time.

5. AN EXTENSION TO KALMANSON MATRICES

In the Introduction, we claimed that our primary motivation in this study is the interest on the biobjective traveling salesman problem with profits (TSPP) under special metrics. In fact, given the equivalence between feasible subtrees and feasible subtours, the approaches suggested above can be used to solve the biobjective TSPP on a complete graph with a tree metric. In this section, we show that such approaches can be extended to solve biobjective TSPP on complete graphs with a Kalmanson metric.

An $n \times n$ distance matrix $C = [c_{ij}]$ is a Kalmanson matrix [24] if the following conditions are satisfied:

$$c_{ij} + c_{kl} \le c_{ik} + c_{jl} \qquad \forall 1 \le i < j < k < l \le n, \tag{8}$$

$$c_{il} + c_{jk} \le c_{ik} + c_{jl} \qquad \forall 1 \le i < j < k < l \le n. \tag{9}$$

These conditions unite two special cases of distance matrices, i.e., trees and convex point sets in the Euclidean plane (Klinz and Woeginger [27]).

Given a complete graph $G = (V,E)$, we identify a Hamiltonian tour in G as a permutation $\pi = \langle i_1, i_2, \ldots, i_n \rangle$ of the vertices in V. A master tour is a Hamiltonian tour π such that, for every $V' \subseteq V$, a minimum traveling salesman tour for V' is obtained by removing from π the vertices not in V'. Deĭneko et al. [9] prove that the permutation $\langle 1, 2, \ldots, n \rangle$ is a master tour if and only if the distance matrix of G is a Kalmanson matrix. Christopher et al. [5] show that checking whether an ordering of vertices exists such that the resulting matrix is a Kalmanson matrix can be done in $O(n^2)$ time.

For convenience, we call KESE and KE the problems corresponding to SubtreeESE and SubtreeE respectively, when the underlying graph has a Kalmanson distance matrix. We will show in some details how to extend the results for SubtreeESE to KESE; the extension from SubtreeE to KE will be just claimed for brevity.

Let $G = (V,E)$ be a graph on the vertex set $V = \{0,1,\ldots,n\}$ with a Kalmanson distance matrix $C = [c_{ij}]$. Let a positive profit p_i be associated with every $i = 1,\ldots,n$ and let $p_0 = 0$. In order to solve KESE, add to G a copy of vertex 0, called $n+1$, and set $c_{0,n+1} = 0$, $c_{i,n+1} = c_{0,i}$ for all $i = 1,\ldots,n$, and $p_{n+1} = 0$. Consider the following parametric program.

$$\begin{aligned}
\min \quad & \sum_{i=0}^{n} \sum_{j=i+1}^{n+1} (\lambda c_{ij} - (1-\lambda) p_i) x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n+1} x_{0j} = 1 \\
& \sum_{j=i+1}^{n+1} x_{ij} - \sum_{j=0}^{i-1} x_{ji} = 0 \qquad (i = 1,\ldots,n+1) \\
& x_{ij} \in \{0,1\} \qquad (i = 0,1,\ldots,n; \\
& \qquad\qquad\qquad j = i+1,\ldots,n+1)
\end{aligned} \tag{10}$$

Every feasible solution of (10) consists in a connected route from 0 to $n+1$ along which the vertex indices are increasing. Because in the original graph G the permutation $\langle 0,1,2,\ldots,n \rangle$ is a master tour, in any efficient subtour the vertices are visited in increasing order of their index. So there is a one-to-one correspondence between feasible subtours in G and feasible solutions of (10).

Model (10) is in fact a parametric shortest path problem on an acyclic network. For any given value of λ, the optimal solution can be found in linear time with respect to the number of arcs [8] which, in our case, corresponds to $O(n^2)$.

Finding the set of optimal solutions for all $\lambda \in [0,1]$ comes down to solving a parametric shortest path problem. This is a hard problem on general graphs; however, for some specific graphs such as series-parallel graphs it is polynomially solvable (Raghavan et al. [33]). We show that also on Kalmanson matrices it is polynomially solvable. Figure 3 represents the network, the cost on each edge (i,j) is given by $d_{ij} = \lambda c_{ij} - (1-\lambda) p_i = \lambda(c_{ij} + p_i) - p_i$. For $\lambda = 0$, the shortest path is $\langle 0,1,2,\ldots,n,n+1 \rangle$ with total cost $C(0) = -\sum_{i=1}^{n} p_i$. For $\lambda = 1$, the shortest path is $\langle 0, n+1 \rangle$, and total cost is $C(1) = c_{0,n+1} = 0$. We state the following.

Lemma 1. *For the parametric shortest path problem in (10) it holds that if some vertex $i \in V$ is not visited in an optimal solution of (10) for $\lambda = \lambda'$, vertex i will also not be visited in any optimal solution of (10) for $\lambda \ge \lambda'$.*

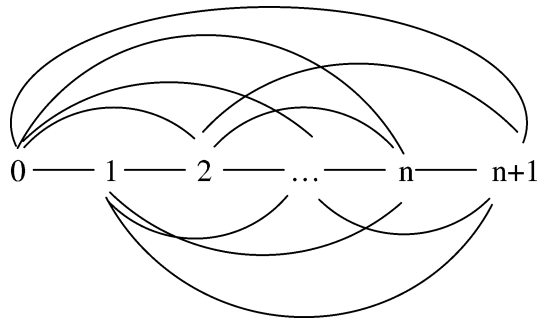


FIG. 3. The network underlying problem (10).

Proof. We start with an observation.

Claim 2. *For increasing values of λ, if a vertex enters the optimal path, then at least another vertex must leave it.*

Claim 2 may be proved as follows. Recall that, for $\lambda = 0$, the optimal path containes all the vertices, in their order. The cost associated with any subpath is a linear function of λ. It is easy to see that the slope associated with an arc $(i, i+k)$ with $k > 1$ is always smaller that the slope associated with any subpath P from i to $i+k$ using some vertices between $i+1$ and $i+k-1$. As a consequence, if for some value λ' the direct arc $(i, i+k)$ becomes cheaper than P then $(i, i+k)$ remains cheaper for any $\lambda \geq \lambda'$. It follows that, for $\lambda \geq \lambda'$, a new optimal path cannot be obtained by simple insertion of vertices of P.

Now we are going to prove the lemma by contradiction. Suppose that Lemma 1 does not hold. Then, a vertex exists which first leaves and then enters the optimal route while λ increases from 0 to 1. For $\lambda = 0$, the optimal path is $\langle 0, 1, 2, \ldots, n+1 \rangle$, visiting all the vertices. While the value of λ increases, vertices drop out of the path. Let ℓ be the first vertex re-entering the path, and let λ^* be the smallest value for λ at which ℓ re-enters the path.

Suppose that for $\lambda' < \lambda^*$ vertex ℓ leaves the optimal path. Because ℓ is the first vertex re-entering the optimal path, at λ' a shortcut is created. Let u' and v' be the left, respectively right, endpoints of such a shortcut. Without loss of generality, we may assume that $u' > 0$ and $v' < n+1$ (possibly including vertices with profit big enough to be the last vertices that leave the optimal path). At this point, the cost for using the direct arc from u' to v' becomes smaller than the cost of the current subpath from u' to v' passing through ℓ. As a consequence, the direct arc is also cheaper than the path $\langle u', \ell, v' \rangle$:

$$\begin{aligned} &\lambda c_{u',v'} - (1-\lambda) p_{u'} \\ &\qquad \leq \lambda (c_{u',\ell} + c_{\ell,v'}) - (1-\lambda)(p_{u'} + p_\ell) \quad \text{for all } \lambda \geq \lambda'. \end{aligned} \tag{11}$$

For $\lambda = \lambda^*$, vertex ℓ enters the path. As a consequence of Claim 2, another vertex will leave the optimal path. We may assume such a vertex is after ℓ. Let P_B and P_A be the optimal paths immediately before, respectively after, λ^*. Let u^* be the greatest vertex before u' included in both P_B and P_A. Let v^* be the smallest vertex after ℓ included in P_B but not in P_A. Finally, let z^* be the smallest vertex after v' included in both P_B and P_A.

The cost of the subpath $\langle u^*, \ell, z^* \rangle$, the only segment where the cost is influenced by ℓ entering and v^* leaving, is given by:

$$\alpha = \lambda^* (c_{u^*,\ell} + c_{\ell,z^*}) - (1-\lambda^*)(p_{u^*} + p_\ell).$$

The direct arc $\langle u^*, z^* \rangle$ would have the following cost:

$$\beta = \lambda^* (c_{u^*,z^*}) - (1-\lambda^*)(p_{u^*}).$$

For ℓ to enter the path, it must thus hold that $\alpha < \beta$. Now, (11) still holds when λ' is replaced by $\lambda^* > \lambda'$. From (11) it follows that:

$$\lambda^* (c_{u',v'}) < \lambda^* (c_{u',\ell} + c_{\ell,v'}) - (1-\lambda^*)(p_\ell)$$

and from $\alpha < \beta$ it follows that

$$\lambda^* (c_{u^*,\ell} + c_{\ell,z^*}) - (1-\lambda^*)(p_\ell) < \lambda^* (c_{u^*,z^*}).$$

Combining both leads to the following equation:

$$c_{u',\ell} + c_{\ell,v'} + c_{u^*,z^*} > c_{u',v'} + c_{u^*,\ell} + c_{\ell,z^*}. \tag{12}$$

We now show that (12) is in contradiction with the Kalmanson conditions. Consider the following Kalmanson conditions:

$$c_{\ell,v'} + c_{u',z^*} \leq c_{u',v'} + c_{\ell,z^*} \tag{13}$$

and

$$c_{u^*,z^*} + c_{u',\ell} \leq c_{u^*,\ell} + c_{u',z^*}. \tag{14}$$

Inequality (13) follows when substituting $u' < \ell < v' < z^*$ for $i < j < k < l$ in (9), and inequality (14) follows when substituting $u^* < u' < \ell < z^*$ for $i < j < k < l$ in (9). The summation over both equations yields:

$$c_{u',\ell} + c_{\ell,v'} + c_{u^*,z^*} \leq c_{u',v'} + c_{u^*,\ell} + c_{\ell,z^*}$$

which is in contradiction with (12). ∎

The optimal value $C(\lambda)$ is a piecewise linear and convex function of the parameter λ. Let $S^\star(\lambda)$ be the optimal set of served vertices for parameter λ; $S^\star$ only changes at the breakpoints of $C(\lambda)$. Thus, to enumerate all supported subtours, it is sufficient to search for the subtours associated with the breakpoints of $C(\lambda)$. We know that $C(0) = -\sum_{i=0}^{n} p_i$ with $S^\star(0) = \{0, 1, 2, \ldots, n+1\}$ the set of served vertices and $C(1) = 0$ with $S^\star(1) = \{0, n+1\}$. Due to Lemma 1 it follows that, for increasing λ, vertices will gradually "drop out" of the path until, for $\lambda = 1$, only the depot remains. Thus, for $\lambda' < \lambda''$, $S^\star(\lambda') \supseteq S^\star(\lambda'')$ and the value of $S^\star$ only changes at $k \leq n$ breakpoints. Summarizing, we have a parametric shortest path problem where $C(\lambda)$ is a linear convex function of λ with $O(n)$ breakpoints. Recall that for any fixed

TABLE 1. A summary of our complexity results.

	Line	Tree	Kalmanson
ESE	$O(n)$	$O(n^2)$	$O(n^3)$
E	$O(n^2 \log n)$	FPTAS	FPTAS

λ, evaluating $C(\lambda)$, i.e., solving problem (10), takes $O(n^2)$ time. Thus, by using the parametric approach from [12], we are lead to the following result.

Theorem 6. *KESE is solvable in $O(n^3)$ time.*

Proof. The proof follows immediately from the previous discussion and the results from [12]. ∎

Concerning the problem of finding all efficient points on a graph with a Kalmanson distance matrix, we just mention, without proof, that KE can be solved by a simple dynamic programming algorithm in $O(n^2 P_{\text{tot}})$ time. From this algorithm, it is possible to develop an FPTAS by following the lines used for SubtreeE.

6. CONCLUSIONS

In this article, we have studied a biobjective combinatorial optimization problem on graphs with a tree metric. We have considered two problems: finding all extreme supported efficient points (SubtreeESE); and finding all efficient points (SubtreeE). For both problems, we have developed efficient algorithms, that are extendable to more general graphs that satisfy the Kalmanson conditions. Moreover, we have analyzed some special cases, including problems on a line. Table 1 summarizes our results.

Acknowledgments

The authors thank the referees for their careful reading of the manuscript and valuable suggestions, and the Associate Editor for pointing out reference [12].

REFERENCES

[1] Y. Asahiro, T. Horiyama, K. Makino, H. Ono, T. Sakuma, and M. Yamashita, How to collect balls moving in the euclidean plane, Electron Notes Theor Comput Sci 91 (2004), 229–245.

[2] I. Averbakh and O. Berman, A heuristic with worst-case analysis for minimax routing of two traveling salesmen on a tree, Discr Appl Math 68 (1996), 17–32.

[3] J. Bérubé, M. Gendreau, and J. Potvin, An exact ε-constraint method for bi-objective combinatorial optimization problems—Application to the traveling salesman problem with profits, Eur J Oper Res 194 (2009), 39–50.

[4] B. Chandran and S. Raghavan, "Modeling and solving the capacitated vehicle routing problem on trees," The vehicle routing problem, B. Golden, S. Raghavan, and E. Wasil (Editors), Springer, New York, NY, 2008, pp. 239–261.

[5] G. Christopher, M. Farach, and M. Trick, The structure of circular decomposable metrics, Proc ESA '96, Lecture Notes in Comput Sci 1136 (1996), 486–500.

[6] S. Coene and F.C.R. Spieksma, Profit-based latency problems on the line, Oper Res Lett 36 (2008), 333–337.

[7] H.W. Corley, Efficient spanning trees, J Optim Theory Appl 45 (1985), 481–485.

[8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to algorithms, 2nd edition, MIT Press, McGraw-Hill, 2001.

[9] V.G. Deĭneko, R. Rudolf, and G.J. Woeginger, Sometimes travelling is easy: The master tour problem, SIAM J Discr Math 11 (1998), 81–93.

[10] M. Ehrgott, Approximation algorithms for combinatorial multicriteria optimization problems, Int Trans Oper Res 7 (2000), 5–31.

[11] M. Ehrgott, Multicriteria Optimization, Springer, Berlin, 2005.

[12] M.J. Eisner and D.G. Severance, Mathematical techniques for efficient record segmentation in large shared databases, J Assoc Comput Mach 23 (1976), 619–635.

[13] T. Erlebach, H. Kellerer, and U. Pferschy, Approximating multiobjective knapsack problems, Manage Sci 48 (2002), 1603–1612.

[14] D. Feillet, P. Dejax, and M. Gendreau, Traveling salesman problems with profits, Trans Sci 39 (2005), 188–205.

[15] P. Friese and J. Rambau, Online-optimization of multi-elevator transport systems with reoptimization algorithms based on set-partitioning models, Discr Appl Math 154 (2006), 1908–1931.

[16] K. Fukuda, Note on new complexity classes $\mathcal{ENP}$, $\mathcal{EP}$ and $\mathcal{CEP}$—an extension of the classes $\mathcal{NP}$, co-$\mathcal{NO}$ and $\mathcal{P}$. Available at http://www.ifor.math.ethz.ch/fukuda/old/ENP_home/ENP_note.html, 1996.

[17] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, Freeman, San Francisco, 1979.

[18] D. Gusfield, Sensitivity analysis for combinatorial optimization, Doctoral Thesis, University of California, Berkeley, 1980.

[19] P. Hansen, "Bicriterion path problems," Multiple criteria decision making: Theory and application, G. Fandel and T. Gal (Editors), Springer-Verlag, Berlin, 1979, pp. 109–127.

[20] M.I. Henig, The shortest path problem with two objective functions, Eur J Oper Res 25 (1985), 281–291.

[21] D.S. Johnson and K.A. Niemi, On knapsacks, partitions, and a new dynamic programming technique for trees, Math Oper Res 8 (1983), 1–14.

[22] N. Jozefowiez, F. Glover, and M. Laguna, Multi-objective meta-heuristics for the traveling salesman problem with profits, J Math Model Algorithms 7 (2008), 177–195.

[23] N. Jozefowiez, F. Semet, and E. Talbi, Multi-objective vehicle routing problems, Eur J Oper Res 189 (2008), 293–309.

[24] K. Kalmanson, Edgeconvex circuits and the traveling salesman problem, Can J Math 27 (1975), 1000–1010.

[25] Y. Karuno, H. Nagamochi, and T. Ibaraki, Vehicle scheduling on a tree with release and handling times, Ann Oper Res 69 (1997), 193–207.

[26] C.P. Keller and M. Goodchild, The multiobjective vending problem: A generalization of the traveling salesman problem, Environ Plan B: Plan Des 15 (1988), 447–460.

[27] B. Klinz and G.J. Woeginger, The steiner tree problem in Kalmanson matrices and in circulant matrices, J Combin Optim 3 (1999), 51–58.

[28] M. Labbé, G. Laporte, and H. Mercure, Capacitated vehicle routing on trees, Oper Res 39 (1991), 616–622.

[29] A. Lim, F. Wang, and Z. Xu, "The capacitated traveling salesman problem with pickups and deliveries on a tree," Proc ISAAC05, Lecture Notes Comput Sci, Beijing, China, 2005, pp. 1061–1070.

[30] I. Muslea, "The very offline *k*-vehicle routing problem on trees," Proc Int Conf Chilean Comput Sci Soc, 1997, pp. 155–163.

[31] C.H. Papadimitriou and M. Yannakakis, "On the approximability of trade-offs and optimal access of web sources," Proc 41st Annu Symp Found Comput Sci, Redondo Beach, CA, 2000, pp. 86–92.

[32] H.N. Psaraftis, M.M. Solomon, T.L. Magnanti, and T. Kim, Routing and scheduling on a shoreline with release times, Manage Sci 36 (1990), 212–223.

[33] S. Raghavan, M.O. Ball, and V.S. Trichur, Bicriteria product design optimization: an efficient solution procedure using and/or trees, Nav Res Logist 49 (2002), 574–592.

[34] V. T'kindt and J. Billaut, Multicriteria scheduling, Springer, Berlin, 2006.

[35] V. T'kindt, K. Bouibede-Hocine, and C. Esswein, Counting and enumeration complexity with application to multicriteria scheduling, 4OR 3 (2005), 1–21.

[36] G.J. Woeginger, When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? INFORMS J Comput 12 (1999), 57–74.