Theoretical Computer Science 531 (2014) 54–65

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Complexity and approximation for Traveling Salesman Problems with profits

Enrico Angelelli [a,*], Cristina Bazgan [b,c], M. Grazia Speranza [a], Zsolt Tuza [d,e,1]

[a] Department of Economics and Management, University of Brescia, Brescia, Italy
[b] Université Paris-Dauphine, LAMSADE, Place du Marechal de Lattre de Tassigny, 75775 Paris Cedex 16, France
[c] Institut Universitaire de France, France
[d] Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary
[e] Department of Computer Science and Systems Technology, University of Pannonia, Veszprém, Hungary

ARTICLE INFO

Article history:
Received 17 May 2011
Received in revised form 30 January 2014
Accepted 24 February 2014
Communicated by T. Erlebach

Keywords:
Routing problem with profit
Computational complexity
Approximation algorithm
Path
Tree
Cycle

ABSTRACT

In TSP with profits we have to find an optimal tour and a set of customers satisfying a specific constraint. In this paper we analyze three different variants of TSP with profits that are NP-hard in general. We study their computational complexity on special structures of the underlying graph, both in the case with and without service times to the customers. We present polynomial algorithms for the polynomially solvable cases and fully polynomial time approximation schemes (FPTAS) for some NP-hard cases.

1. Introduction

A huge number of papers has studied the well known Traveling Salesman Problem (TSP). In the TSP all customers are to be visited. There exist, however, several practical situations where customers have to be selected among a set of potential customers on the basis of their profits. The goal is to find an appropriate trade-off between the total collected profit and the cost of the tour. A survey by Feillet et al. [13] defines these problems as the *Traveling Salesman Problems with profits*. The objective function may be the difference between the total profit collected and the cost of the tour (Profitable Tour Problem or PTP), the maximization of the total collected profit (Orienteering Problem or OP), the minimization of the cost of the tour (Prize-Collecting TSP or PCTSP). These problems are all NP-hard in general. To further enlighten the computational complexity of these problems, we investigate their complexity on some classes of instances, characterized by

- customers located on a path exiting from the depot;
- customers located on a cycle;

* Corresponding author.
E-mail addresses: angele@eco.unibs.it (E. Angelelli), bazgan@lamsade.dauphine.fr (C. Bazgan), speranza@eco.unibs.it (M.G. Speranza), tuza@dcs.uni-pannon.hu (Z. Tuza).

[1] Research supported in part by the Hungarian Scientific Research Fund, OTKA grant T-81493.

http://dx.doi.org/10.1016/j.tcs.2014.02.046
0304-3975/

- customers located on a star rooted at the depot;
- customers located on a tree rooted at the depot.

These classes of instances correspond to special structures of the underlying graph, namely a path, a cycle, a star, a tree.

We study the computational complexity of the problems on these classes of instances, both in the case a service time is associated with each customer and in the case without service times. We present polynomial algorithms for the polynomially solvable cases and fully polynomial time approximation schemes for the NP-hard cases. We also present a reduction to eliminate service times in general networks.

In the rest of the introduction we define the problems we study, present earlier results and summarize the results of this paper.

In Section 2 we prove that the TSP with profits we study are intractable already on some rather restricted classes of instances. These results justify the interest in some even more restricted network topologies. Moreover, at the end of this section, we present a reduction lemma which transforms a problem instance with service times to one without. In Section 3 we prove solvability in linear time of four cases from which we derive the linear time solvability of a number of special cases. Fully polynomial time approximation schemes are presented in Section 4 for OP and PCTSP on tree networks that can be directly applied to star and path networks and easily adapted to cycle networks.

1.1. Problem definitions

We assume that a road network is represented by a connected weighted graph $G = (V, E)$ where the vertices $V = \{i \mid i = 0, \ldots, n\}$ represent road intersections and locations of relevant points like the depot and the customers. We assume throughout that the depot is indexed by 0 and other vertices are indexed from 1 to n. The edges represent physical connections between pairs of vertices (i, j) and their weights $t_{i,j}$ represent the traveling time from vertex i to vertex j. In general, visiting a customer i requires a service time $s_i \geqslant 0$ and offers a profit $p_i \geqslant 0$; nevertheless, a customer vertex in the network can be traversed without visiting it. This means that no service time is spent and no profit is collected. For sake of uniformity we consider road intersection vertices as customers with no profit and zero service time. In this paper we assume all parameters $t_{i,j}$, s_i and p_i to be integers.

The problem can be formulated in different versions. In all cases a subset of customers $S \subseteq \{1, \ldots, n\}$ has to be selected in order to optimize different objective functions and satisfy different constraints involving both the total profit $P_S = \sum_{i \in S} p_i$ and the smallest possible time, denoted by TSP_S, of visiting all the customers in S, which is calculated from the minimum time length of a tour starting from (and returning to) the depot plus the total service time $ST_S = \sum_{i \in S} s_i$.

We recall the different variants of the problem as reported in the literature (see, for instance, Feillet et al. [13]).

Profitable Tour Problem – PTP

$$\max(P_S - TSP_S)$$

Orienteering Problem – OP

$$\max(P_S)$$

such that

$$TSP_S \leqslant T_{\max}$$

where $T_{\max}$ is given.

Prize Collecting TSP – PCTSP

$$\min(TSP_S)$$

such that

$$P_S \geqslant P_{\min}$$

where $P_{\min}$ is given.

It is worth noticing that there is not a general agreement in the literature about problem naming and formulation. Namely, what is called Profitable Tour Problem (PTP) by Feillet et al. [13] is called Prize-Collecting problem by authors like Bienstock et al. [7]. Moreover, PTP is sometimes formulated as a minimization problem: $\min(TSP_S + P_{\bar{S}})$ trying to find a tour that minimizes the cost and the profit missed for not visiting vertices in S. As pointed out in Johnson et al. [18], the two formulations are equivalent as optimization problems (complementing each other), but they are not when approximation algorithms are concerned. From now on, in this paper, we follow the terminology proposed above and refer to the minimization formulation of PTP as $\text{PTP}_{\min}$.

1.2. Earlier results for connected networks

Note that given any connected network represented by a graph $G' = (V', E')$ an equivalent complete graph $G = (V, E)$ can be derived.

All problems are obviously NP-hard (see Feillet et al. [13]). Some results are known that concern the possibility to find approximation algorithms with guaranteed performance.

Bienstock et al. [7] propose a polynomial-time approximation algorithm having a 2.5 performance guarantee for the PTP_{min}. Goemans and Williamson [17] improve the above performance guarantee and obtain a $(2-1/(n-1))$-approximation algorithm with a purely combinatorial approach.

Arkin et al. [2] give a 2-approximation for the rooted OP for the special case of points that have all the same prize and are distributed in the Euclidean plane. They find 2 and 3-approximation algorithms when the required output is a path or a tree, respectively. Bansal et al. [6] obtain a 3-approximation for a more constrained version of OP called "point-to-point OP", in which the starting location s as well as the ending location t are fixed. The OP cannot be approximated in polynomial time to a factor better than 1481/1480, unless P = NP (Blum et al. [8]). Recently, Chekuri et al. [11] studied the point-to-point OP on directed and undirected graph with time windows. The goal is still to find a walk in the graph which maximizes the number of visited vertices and satisfying a budget constraint. They propose a $(2+\varepsilon)$-approximate algorithm that improves on existing literature.

Awerbuch et al. [5] derive an approximation algorithm for the PCTSP from an approximation algorithm for the k-minimum-spanning-tree problem (k-MST problem). Their approximation algorithm achieves a factor $O(\log^2(\min(n, P_{\text{min}})))$ for the PCTSP. Several other papers have found approximation results for the k-MST problem, and consequently for the PCTSP (see Blum et al. [9], Arora and Karakostas [4], Arora [3], Garg [15]). It seems the best known result is a 2-approximation due to Garg [16]. Most of these results are based on a classic primal–dual algorithm for PCTSP due to Goemans and Williamson [17].

In literature, there is also a stream of research focusing on the Prize-Collecting Steiner Tree Problem (PCSTP). A Steiner tree on a graph $G = (V, E)$ is a tree spanning a fixed subset S of the vertices V. This kind of problems arises in fields where the aim is to create a network connecting relevant points rather than creating routes servicing a set of points from a depot. When the vertices in S produce profits, but all of them cannot be feasibly or optimally connected due to budget constraints or cost penalties, the Steiner tree problem gives rise to variants strongly connected to the problems we are dealing with in this paper. Indeed, for particular graph structures like paths, stars and trees, a tour is obtained by doubling a sub-tree of the original graph. Even in the case of a circle, besides the possibly feasible tour visiting all vertices clockwise, other tours are obtained by doubling a tree with at most two main branches. In order to use a Steiner tree problem algorithm to solve the tour problem on a tree we should halve the traveling times across the edges in the original graph because to have a tour each edge has to be traversed twice. We briefly report some recent references to the literature on Steiner tree problems. Klau et al. [20] study a problem where an optimal sub-tree is required on a tree of n vertices and, quite originally, the objective function is computed as the ratio between profits and edge costs. For this problem they propose an $O(n \log n)$-time algorithm. On their way, they also propose a linear time algorithm to solve PTP on trees without service time. Feofiloff et al. [14] slightly improve on the $(2-1/(n-1))$-approximation in [17] by a $(2-2/n)$-approximation algorithm running in $O(n^2 \log n)$ time. Chekuri et al. [10] propose a PTAS for the PTP_{min} in planar graphs, hence working on trees as well. Archer et al. [1] solve the unrooted PTP_{min} providing $(2-\varepsilon)$-approximation algorithms when a tree/cycle/path subgraph is required in output. A survey can be found in Costa et al. [12]. However we stress the fact that all these results are obtained for general input graphs, in our work we address problems where the input graph structure is a path/star/tree/cycle and the required output is a tour.

1.3. Our results

The contribution of this paper is the classification of PTP, OP and PCTSP according to their computational complexity in a 2-dimensional operational settings. The first dimension accounts for the topology of simple networks like stars, cycles, paths and trees. The second dimension considers the presence of service times versus the alternative scenario where all service times are zero.

It will turn out that PTP is solvable in linear time on all considered topologies and service settings. On the other side, OP and PCTSP are both linearly solvable on linear topologies (paths and cycles) without service times, but become NP-hard as the topology moves to the simplest tree (a star) or service times are introduced.

We propose a set of pseudo-polynomial algorithms for OP with service times whose complexity change according to the network topology. Namely, given P as the sum of all profits, we can solve OP in $O(nT_{\max})$ or $O(nP)$ time on path networks, $O(n^2 T_{\max})$ or $O(n^2 P)$ time on cycle networks and $O(nT_{\max}^2)$ or $O(nP^2)$ time on tree networks.

The algorithms will be formulated after a transformation of trees into trees without service times, paths into special trees without service times, and cycles into a set of problems on paths.

Eventually, we propose an FPTAS that approximates within $(1-\varepsilon)$ instances with service times on trees, cycles and paths in $O(n^5/\varepsilon^2)$, $O(n^4/\varepsilon)$ and $O(n^3/\varepsilon)$ time, respectively. An FPTAS on cycles is discussed as corollary of Theorem 12.

Obviously, all pseudo-polynomial algorithms and FPTAS can be applied to problems without service times.

Table 1
Problem complexity, where $P = \sum p_i$, $T = \sum t_{i,j}$.

		PTP		OP		PCTSP	
		No_ST	With_ST	No_ST	With_ST	No_ST	With_ST
Cycle	complexity				[NP-hard]		[NP-hard]
	time	$[O(n)]$	$O(n)$	$O(n)$	$O(n^2 T_{\max})$, $O(n^2 P)$	$O(n)$	$O(n^2 P_{\min})$, $O(n^2 T)$
Path	complexity				NP-hard		NP-hard
	time	$[O(n)]$	$[O(n)]$	$[O(n)]$	$O(nT_{\max})$, $O(nP)$	$[O(n)]$	$O(nP_{\min})$, $O(nT)$
Star	complexity			NP-hard	[NP-hard]	NP-hard	[NP-hard]
	time	$[O(n)]$	$[O(n)]$				
Tree	complexity			[NP-hard]	[NP-hard]	[NP-hard]	[NP-hard]
	time	$[O(n)]$	$O(n)$		$O(nT^2_{\max})$, $O(nP^2)$		$O(nP^2_{\min})$, $O(nT^2)$

Table 2
FPTAS complexity, where $P = \sum p_i$, $T = \sum t_{i,j}$.

	FPTAS	
	OP $(1-\varepsilon)$-approx	PCTSP $(1+\varepsilon)$-approx
Path with service times	$O(\frac{n^3}{\varepsilon})$	$O(n^2(\ln T)/\varepsilon)$
Cycles with service times	$O(\frac{n^4}{\varepsilon})$	$O(n^3(\ln T)/\varepsilon)$
Tree with service times	$O(\frac{n^5}{\varepsilon^2})$	$O(n^3(\ln T)^2/\varepsilon^2)$

Problem PCTSP shares the same type of complexity with OP. In particular, given T as the sum of all edge traversing times, we show that the problem can be solved in $O(nP_{\min})$ or $O(nT)$ time on paths, in $O(n^2 P_{\min})$ or $O(n^2 T)$ time on cycles and in $O(nP^2_{\min})$ or $O(nT^2)$ time on trees. Furthermore an FPTAS is proposed for PCTSP so that tree, cycle and path instances can be approximated within $(1+\varepsilon)$ in $O(n^3(\ln T)^2/\varepsilon^2)$, $O(n^3(\ln T)/\varepsilon)$ and $O(n^2(\ln T)/\varepsilon)$ time, respectively.

The results are summarized in Tables 1 and 2. We present the approximation ratio α as a constant such that $H \leqslant \alpha \cdot OPT$ for minimization problems, and $H \geqslant \alpha \cdot OPT$ for maximization problems, where H and OPT are the heuristic and optimal value, respectively. Please note that we get $\alpha \geqslant 1$ for minimization problems and $\alpha \leqslant 1$ for maximization problems. Our notation for the maximization problems differs from the one used in the above cited literature. The cited approximation ratios have to be interpreted in our notation as $1/\alpha$.

In Table 1 we report the complexity of the problems for different networks and service time settings. Results not embraced in brackets are directly proved in the paper with the exception of case of PTP on trees that has been solved by [20] in the more general framework of Steiner trees. Those between brackets are derived from results proven either by restrictions (in case of polynomial complexity) or generalizations (in case of NP-hardness). Inheritance flow for algorithm complexity is: Tree → Path, Tree → Star, Cycle → Path, With_ST → No_ST. Inheritance flow for NP-hardness is the reverse.

In Table 2 the complexity of approximation algorithms proposed in the paper for the NP-hard cases is summarized.

2. NP-hardness and service time reduction

In this section we prove that the three problems defined above, PTP, OP, and PCTSP, are intractable already for some rather restricted classes of instances. These results justify why some even more restricted network topologies should be studied in detail, as it will be done in later sections. From another point of view, at the end of this section we present a reduction lemma which transforms a problem instance with service times to one without, with a very little change in the structure of network.

It will turn out that OP and PCTSP represent similar complexity level, but different from PTP. Therefore here we deal with the former two, and prove that they are intractable already on rather restricted instances. In the NP-hardness reductions we shall apply the following two basic optimization problems.

MAX KNAPSACK
Input: A set S of n items where each item i has value v_i and weight w_i, $i = 1, \ldots, n$, and an integer B.
Output: A subset $S' \subseteq S$ such that $\sum_{i \in S'} w_i \leqslant B$ and such that $\sum_{i \in S'} v_i$ is maximized.

MIN KNAPSACK
Input: A set S of n items where each item i has value v_i and weight w_i, $i = 1, \ldots, n$, and an integer B.
Output: A subset $S' \subseteq S$ such that $\sum_{i \in S'} w_i \geqslant B$ and such that $\sum_{i \in S'} v_i$ is minimized.

Both of these problems are well known to be NP-hard [19].

2.1. Tree networks without service times

Here we show NP-hardness of OP and PCTSP on trees. In fact, the following even more restricted result is valid.

Theorem 1. OP *and* PCTSP *are NP-hard even on stars without service times.*

Proof. For OP we construct a polynomial reduction from the decision version associated to Max Knapsack problem. Given an input I of Max Knapsack with n items where each item i has value v_i and weight w_i, $i = 1, \ldots, n$ and two integers b and k, we construct an instance I' of OP as follows: we consider a star with the center 0 and leaves $1, \ldots, n$. The traveling time from 0 to i is w_i and the profit of customer i is v_i. It is clear that there exists a subset $S' \subseteq S$ such that $\sum_{i \in S'} w_i \leqslant b$ and such that $\sum_{i \in S'} v_i \geqslant k$ if and only if there exists a tour visiting 0 and S' of time-length at most $2b$ and profit at least k.

For PCTSP one can apply a similar argument starting from Min Knapsack. □

2.2. Path and cycles networks with service times

Here we prove that service times make OP and PCTSP hard already on the simplest class of network topology.

Theorem 2. OP *and* PCTSP *are NP-hard on paths with service times.*

Proof. Similarly to Theorem 1, it suffices to give detailed proof for OP. We construct a polynomial reduction from the decision version associated to Max Knapsack problem. Given an input I of Max Knapsack with n items where each item i has value v_i and weight w_i, $i = 1, \ldots, n$ and two integers b and k, we construct an instance I' of OP as follows: we consider a path on $n+1$ vertices with edges $(i, i+1)$, $i = 0, \ldots, n-1$. The traveling time from i to $i+1$ is 0, the service time of customer i is w_i and the profit of customer i is v_i. It is clear that there exists a subset $S' \subseteq S$ such that $\sum_{i \in S'} w_i \leqslant b$ and such that $\sum_{i \in S'} v_i \geqslant k$ if and only if there exists a tour starting from 0 and visiting exactly S' of time-length at most b and profit at least k. □

Corollary 3. OP *and* PCTSP *are NP-hard on cycles with service times.*

Proof. The cycle network reduces to a path when one edge entering the depot has traveling time $+\infty$. □

2.3. Service times reduction

Here we prove another kind of reduction, which describes a complexity equivalence between instances with and without service times. Assume an instance $G = (V, E)$, a connected graph with $V = \{0, 1, \ldots, n\}$ where 0 represents the depot, the edges $(i, j) \in E$ are assigned with time length $t_{i,j}$ for $0 \leqslant i, j \leqslant n$, and the service time and customer profit at i is s_i and p_i, respectively, for $i = 1, \ldots, n$. For each customer i where the inequality $s_i > 0$ holds, the network graph will be augmented by creating a new vertex i' and connected to the graph according to the following procedure.

Definition 4. Given a problem instance G above, we define the *leaf extension* $G^+ = (V^+, E^+)$ of G as the problem instance without service times, derived from G in the following way. Let

- $V^+ = V \cup \{i' \mid s_i > 0\}$,
- $E^+ = E \cup \{(i, i') \mid s_i > 0\}$,

and assign the following costs for all $i' \in V^+ \setminus V$:

- set $s_{i'} = 0$, $p_{i'} = p_i$, $t_{i,i'} = s_i/2$,
- redefine $s_i = 0$, $p_i = 0$.

All $t_{i,j}$ remain unchanged for $(i, j) \in E$, and so do the p_i for all i whose original service time is zero.

Clearly, for every problem instance G, the leaf extension G^+ can be constructed in linear time.

Lemma 5 *(Service Times Reduction Lemma). There is a one-to-one correspondence between the optimal solutions of G and G^+. Moreover, the elimination of service times takes linear time.*

Proof. After the augmentation of the original network all the service times are transformed in virtual traveling times and the number of vertices is not more than $2n + 1$. Profit at i has to be collected if and only if the corresponding tour in G^+ visits i', no matter whether or not the tour in G passes through i. □

The class of trees is closed with respect to leaf extension transformation, but the classes of paths and cycles are not. In particular, leaf extension changes the topology of a path producing a tree. We shall see in Section 3.2 that the exclusion of service times on paths yields indeed substantial decrease in complexity as opposed to the intractability results of the previous subsection.

3. Exact algorithms in linear time

We prove solvability in linear time of PTP, OP, and PCTSP for a number of special cases. In particular we directly prove linear time solvability of PTP on trees with service times, PTP on cycles with service times and eventually of OP and PCTSP on cycles without service times (we already know that OP and PCTSP with service times are NP-hard on general networks).

3.1. Tree networks

In this subsection we briefly discuss the complexity of PTP on trees, stars and paths.

Theorem 6. PTP *can be solved in* $O(n)$ *time on trees with service times.*

The property follows quite straightforwardly from the work of Klau et al. [20] where they prove that the corresponding problem on Steiner tree is solvable in linear time. Our problem can be transformed in an instance of the problem solved in [20] by replacing any profitable node i with service time s_i with a dummy node t_i with null profit and null service time. Node i is then appended as a child to node t_i with edge cost equals to $s_i/2$.

Note that, problems on stars and paths with service times can be seen as special cases of trees with service times. Furthermore, any problem on a star, a path or a tree without service times is a special case of the corresponding problem with service time (set $s_i = 0$ for all customers i) and can be solved by the algorithm for trees with service times. Thus, we have the following corollary.

Corollary 7. PTP *can be solved in* $O(n)$ *time also on trees without service times and on stars and paths with/without service times.*

3.2. Cycle networks

Contrary to PTP, we have seen that OP and PCTSP are NP-hard already on stars without service times, and also on paths and cycles with service times. Hence we assume here that the network is a cycle and for OP and PCTSP the condition $s_i = 0$ holds for all customers.

Theorem 8. *The following problems can be solved in* $O(n)$ *time on cycles:* PTP *with service times,* OP *and* PCTSP *without service times.*

Proof. We assume that the indexing along the cycle is $0, 1, 2, \ldots, n, 0$ in this order, and every tour starts from, and returns to, 0. There can occur two essentially different types of a tour:

- Type 1 — all around the cycle;
- Type 2 — visiting vertices $1, \ldots, i$ in one direction, returning to 0, then in the other direction visiting $n, n-1, \ldots, j+1, j$ and returning to 0 via n.

We shall refer to the second case as the (i, j)-*tour*, where $0 \leqslant i < j \leqslant n+1$. By label $n+1$ we mean 0, and in this way $(i, n+1)$ is interpreted as a tour not using the edge joining 0 and n at all, hence only going from 0 to i and back.

Observe that Type 1 has a unique value for PTP, which can be expressed as

$$\sum_{i=1}^{n} p_i - \sum_{i=0}^{n} t_{i,i+1} - \sum_{i=1}^{n} s_i$$

where $t_{n,n+1}$ means $t_{n,0}$ by the convention put on label $n+1$. Moreover, the Type 1 tour is feasible for OP if $\sum_{i=0}^{n} t_{i,i+1} \leqslant T_{\max}$, and it is feasible for PCTSP if $\sum_{i=1}^{n} p_i \geqslant P_{\min}$. (In OP and PCTSP the summation of the s_i is zero as service times have been excluded.)

In either case, the Type 1 value computed above will have to be compared with the Type 2 optimum, i.e. with the best possible choice of an (i, j)-tour. Since there are $\binom{n+1}{2}$ choices for the pair i, j, the main point is how to go down from $O(n^2)$ to $O(n)$ with proper organization of computation. To achieve this, we propose a linear-time preprocessing in two phases, and then a one-round scanning, as described below.

Algorithm for PTP For PTP, w.l.o.g., we assume that $s_i \leqslant p_i$ for all i. If there would be some i with $s_i > p_i$, then we can consider that vertex i is never visited. First we compute the profit of a tour from 0 to each vertex in both directions:

$$P_i = \sum_{1 \leqslant k \leqslant i} (p_k - 2t_{k-1,k} - s_k), \qquad P'_j = \sum_{j \leqslant k \leqslant n} (p_k - 2t_{k,k+1} - s_k)$$

Accordingly, $P_0 = P'_{n+1} = 0$. Since a farther vertex is of interest only if it yields larger profit, during this procedure we also mark i with $*$ (j with $*'$) if P_i (P'_j) is larger than all preceding calculated values in that direction. Computing all P_i and P'_j requires only $O(n)$ steps since the difference between two consecutive values is expressed with just three terms. Moreover, storing the actual largest profit in a buffer, the condition of marking can be tested in constant time per vertex. Note that 0 is marked with both $*$ and $*'$ because, although it has 0 profit, it has not been preceded with any value.

At the end of this phase each vertex may have one, or both, or none of the markings $*$ and $*'$. We need one more scanning around the cycle, during which we create a list L of the pairs (i, j) with $0 \leqslant i < j \leqslant n$ satisfying the following properties: i is marked with $*$, j is marked with $*'$, and the set $S_{i,j} = \{\ell \mid i < \ell < j\}$ contains no marked vertices at all. (If i, j have the required marks and $j = i + 1$, then $S_{i,j} = \emptyset$ and consequently $(i, j) \in L$.) Then the optimum for PTP is equal to the largest of $\max_{(i,j)\in L}(P_i + P'_j)$ and of the Type 1 tour.

Algorithm for OP First determine the largest value i_0 of index i reachable within $T_{\max}/2$ from the depot (i.e., it requires at most $T_{\max}$ time to travel from 0 to i_0 and back to 0). Then for each i from i_0 down to 0 find the largest value j_i of index j such that j_i is feasible under the assumption that i has been visited. Denoting by $OP^+(i)$ and $OP^-(j)$ the profits of a tour from 0 to i with increasing index and from $n + 1$ to j with decreasing index, respectively (both returning to 0), the maximum OP-value is reached by

$$\max_{0 \leqslant i \leqslant i_0} OP^+(i) + OP^-(j_i).$$

This can be determined in at most $i_0 + (n - j_0 + 1) + 1 \leqslant 2n + 1$ steps by moving with both indices in increasing order, along the sequence of pairs $(0, j_0), (1, j_0), (1, j_0 - 1), \ldots, (1, j_1), (2, j_1), \ldots, (2, j_2 + 1), (2, j_2), \ldots, (i_0, j_{i_0})$, checking in each step whether the current (i, j) meets the requirements. (If also $(i + 1, j_i)$ is feasible for some i, then $j_{i+1} = j_i$ holds.) Local computation requires constant time, hence the algorithm is linear.

Algorithm for PCTSP The method is similar to that for OP, except that now we need to choose i_0 as smallest i yielding $P_{\min}$ on $1, \ldots, i$. Then for i running from i_0 to 0 find the largest $j = j_i$ such that j is feasible under the assumption that i has been visited. Maximum PCTSP-value is reached by choosing largest sum of values at i and j for an (i, j)-tour. The reason for linear running time is similar to the previous case. □

Since problems without service times are particular cases of problems with service times (as already shown in Corollary 7), and problems on paths are particular cases of the corresponding problems on cycles (add an edge of cost $+\infty$ between the endpoints of the path) we have the following corollary.

Corollary 9. *The following problems can be solved in $O(n)$ time:* PTP *on cycles without service times, and* OP *and* PCTSP *on paths without service times.*

4. Pseudo-polynomial algorithms and fully polynomial-time approximation schemes

In this section we design efficient approximation algorithms for OP and PCTSP on tree networks that can be directly applied to star networks, path networks, and can be easily adapted to cycle networks. First of all, let us recall that Lemma 5 yields the algorithmic equivalence of problems with and without service times, whenever the class of input is closed under attaching pendant edges. Because of this reason, for trees it suffices to restrict our attention to the case where $s_i = 0$ holds for all i.

We consider the depot as the root of the tree, with index $i = 0$. For any vertex i we denote by $d_i \geqslant 0$ the number of its children. The children of vertex i are denoted by i_k, where $1 \leqslant k \leqslant d_i$.

Let us observe further that a polynomial-time transformation of the tree can lead to an equivalent *binary tree network*, equivalent in the sense that for any subset of customers in the original tree exactly the same visiting costs and profit is obtained also on the correspondent binary tree. Using the notation introduced at the beginning of Section 3.1, the transformation proceeds as follows:

- If a vertex i has more than 2 children ($d_i > 2$), then the connection between i and its children is removed and two new vertices i' and i'' with zero profit (and zero service time) are generated. Vertex i' is inserted between vertex i and its children i_1 and i_2, while vertex i'' is inserted between vertex i and all its remaining children i_k ($2 < k \leqslant d_i$). The $2 + d_i$ new edges are weighted as follows: $t_{i,i'} = t_{i,i''} = 0$, $t_{i',i_k} = t_{i,i_k}$ for $k = 1, 2$ and $t_{i'',i_k} = t_{i,i_k}$ for $k = 3, \ldots, d_i$.

The transformation can be applied by traversing the dynamically updated tree with any order and requires no more than $n/2$ updates with the insertion of 2 new nodes at each update. The final number of nodes in the tree cannot be larger than $2n$.

Two-step tree reduction Based on the above facts, a general problem instance with a tree network can be reduced in two steps to a simpler instance, as follows.

1. Apply the Service Times Reduction Lemma.
2. Transform the tree to a binary tree.

These steps substantially simplify structure and improve the running time of algorithms.

4.1. Orienteering problem

We design algorithms with various time bounds, their comparison depends on the travel cost and profit conditions. Due to its better time bound, we include the case of paths in a separate assertion.

Theorem 10. OP *can be solved on trees with service times within the following pseudo-polynomial time bounds:*

(*i*) $O(nT_{\max}^2)$.
(*ii*) $O(nP^2)$, *where* $P = \sum_{i=1}^{n} p_i$.
(*iii*) $O(nT_{\max})$ *if the input is a path.*
(*iv*) $O(nP)$ *if the input is a path.*

Proof. In all cases the beginning of the proposed algorithms consists in applying the two-step tree reduction so that we have to deal with a tree of at most $2n$ customers without service times.

Proof of case (*i*). For each vertex $i \geqslant 0$ compute a function value $z_i(t)$ for $t \leqslant T_{\max}$. Function values $z_i(t)$ are computed traversing the tree in post-order, representing the maximum profit that can be collected from the subtree rooted in i within a given time bound t. Function values are $-\infty$ when $t < 0$, and can be stored in a vector with $T_{\max} + 1$ components for $t = 0, \ldots, T_{\max}$. Vector components are computed as (p_i and d_i are the profit and the number of children for vertex i, respectively)

$$z_i(t) = \begin{cases} p_i & \text{if } d_i = 0; \\ p_i + \max(0, z_{i_1}(t - 2t_{i,i_1})) & \text{if } d_i = 1; \\ p_i + \max \begin{cases} 0 \\ z_{i_1}(t - 2t_{i,i_1}) \\ z_{i_2}(t - 2t_{i,i_2}) \\ \max\limits_{2t_{i,i_1} \leqslant \ell \leqslant t - 2t_{i,i_2}} (z_{i_1}(\ell - 2t_{i,i_1}) + z_{i_2}(t - \ell - 2t_{i,i_2})) \end{cases} & \text{if } d_i = 2. \end{cases}$$

The optimal value of the problem is $z_0(T_{\max})$. The optimal customer subset can be retrieved by storing for each $z_i(t)$ the children (if any) and their corresponding time bounds which have given positive contribution to the function value.

Post-order traversal requires $O(n)$ steps. At nodes i with $d_i < 2$ each component is computed in $O(1)$ time and the whole vector z_i is computed in $O(T_{\max})$ time. At nodes i with $d_i = 2$ an $O(T_{\max})$ time is needed for each component and the whole vector z_i is computed in $O(T_{\max}^2)$ time. The maximum number of nodes with 2 children is $2n/2$, thus all vectors z_i are computed in $O(nT_{\max}^2)$ time.

Proof of case (*ii*). For each vertex i we compute a function value $z_i(p)$ for $p = 0, \ldots, P$ that can be stored in a vector with $P + 1$ components. Here $z_i(p)$ represents the minimum time needed for collecting profit exactly p in the subtree rooted in node i. We consider $z_i(p) = +\infty$ if $p < 0$. Vector components are computed as

$$z_i(p) = \begin{cases} 0 & \text{if } p = p_i; \\ +\infty & \text{if } p < p_i; \\ +\infty & \text{if } d_i = 0 \text{ and } p \neq p_i; \\ 2t_{i,i_1} + z_{i_1}(p - p_i) & \text{if } d_i = 1; \\ \min \begin{cases} 2t_{i,i_1} + z_{i_1}(p - p_i) \\ 2t_{i,i_2} + z_{i_2}(p - p_i) \\ 2t_{i,i_1} + 2t_{i,i_2} + \min\limits_{1 \leqslant \ell \leqslant p - p_i - 1} (z_{i_1}(\ell) + z_{i_2}(p - p_i - \ell)) \end{cases} & \text{if } d_i = 2. \end{cases}$$

The optimum corresponds to the maximum value of p such that $z_0(p) \leqslant T_{\max}$. Computing $z_i(p)$ for $d_i = 2$ takes $O(p)$ steps, therefore the running time of the algorithm is $O(nP^2)$.

Proof of case (*iii*). Note that, after the two-step tree reduction, nodes i with two children ($d_i = 2$) have the property that at least one of the two children is a leaf, which we assume to be child i_1 without loss of generality. Thus, the recursion

$$\max_{2t_{i,i_1} \leqslant \ell \leqslant t-2t_{i,i_2}} \left(z_{i_1}(\ell - 2t_{i,i_1}) + z_{i_2}(t - \ell - 2t_{i,i_2})\right)$$

of the algorithm can be reduced to

$$p_{i_1} + z_{i_2}\left([t - 2t_{i,i_1}] - 2t_{i,i_2}\right);$$

indeed if we visit both children, then the traveling salesman will spend exactly $2t_{i,i_1}$ time visiting the leaf i_1 and the rest of the time visiting the subtree rooted in i_2. As a consequence, computational time spent in this step is $O(1)$ instead of $O(T_{\max})$, vector z_i can be filled in $O(T_{\max})$ and the overall cost of the algorithm is $O(nT_{\max})$.

Proof of case (*iv*). Similarly to case (*iii*) we change the recursion

$$2t_{i,i_1} + 2t_{i,i_2} + \min_{1 \leqslant \ell \leqslant p-p_i-1} \left(z_{i_1}(\ell) + z_{i_2}(p - p_i - \ell)\right)$$

to

$$2t_{i,i_1} + 2t_{i,i_2} + z_{i_2}\left([p - p_{i_1}] - p_i\right);$$

also in this case the time spent to compute $z_i(p)$ reduces from $O(P)$ to $O(1)$ and the overall complexity is $O(nP)$.

This completes the proof of Theorem 10. □

Theorem 11. OP *can be solved on cycles with service times within pseudo-polynomial time* $O(n^2 T_{\max})$ *and* $O(n^2 P)$ *where* $P = \sum_{i=1}^{n} p_i$.

Proof. Similarly to the notation used in the proof of Theorem 8, we assume that the indexing along the cycle is $0, 1, 2, \ldots, n, 0$ in this order, and every tour starts from, and returns to, 0. There can occur two essentially different types of a tour:

- Type 1 — all around the cycle;
- Type 2 — passing through vertices $1, \ldots, i$ in one direction, returning to 0, then in the other direction passing through $n, n-1, \ldots, j+1$, $j > i$ and returning to 0 via n.
  Note that in this case we do not assume that passing through a customer vertex implies a visit to the customer spending the corresponding service time.

Let us define Z_{-1} as the optimal solution value that can be obtained by a Type 1 tour and define Z_i $(i = 0, 1, \ldots, n)$ as the optimal solution value that can be obtained by a Type 2 tour reaching customer i as the farthest customer on the first round.

Evaluating Z_{-1} is equivalent to solve a Max Knapsack problem where item values are given by customer profits p_i, item weights are given by service times s_i and maximum weight is given by $T_{\max} - T$ (where $T = \sum_{i=0}^{n} t_{i,i+1}$). This can be done by dynamic programming in $O(nT_{\max})$ and $O(nP)$ time.

Evaluating Z_i $(i = 0, \ldots, n)$ is equivalent to solve a path instance where the cycle network is transformed as follows. Edge $(i, i+1)$ is removed. Customers $j < i$ are relocated to the location of customer i. Customer n is detached from the depot and reconnected to customer i so that the new edge has traveling time equal to the original edge $(0, n)$. According to Theorem 10, computing Z_i takes at most $O(nT_{\max})$ and $O(nP)$.

The problem can be solved taking the maximum over Z_h for $h = -1, 0, 1, \ldots, n$, that is in $O(n^2 T_{\max})$ and $O(n^2 P)$. □

Theorem 12. OP *has FPTAS on trees with service times. For any given* $\varepsilon > 0$, *there is a* $(1-\varepsilon)$*-approximation algorithm with running time* $O(n^5/\varepsilon^2)$ *on trees and* $O(n^3/\varepsilon)$ *on paths.*

Proof. Consider an instance I of OP on a tree with n vertices with traveling times $t_{i,j}$ and profits p_i. In order to obtain a $(1-\varepsilon)$-approximation for I we consider the instance I' on the same tree and with the same traveling times $t_{i,j}$ but with the profits scaled as follows: $p'_i = \lfloor \frac{p_i}{f} \rfloor$ where the factor f will be given later. Thus, $p_i/f - 1 < p'_i \leqslant p_i/f$. Denote by S (S') an optimal solution for I (I'). We prove in the following that an optimal solution for I' obtained by the dynamic programming algorithm given in Theorem 10(*ii*) is a $(1-\varepsilon)$-approximation for I in time polynomial in $|I|$ and $1/\varepsilon$. The value of the solution S' in I is $val(S') = \sum_{i \in S'} p_i \geqslant f \sum_{i \in S'} p'_i \geqslant f \sum_{i \in S} p'_i \geqslant f \sum_{i \in S} (\frac{p_i}{f} - 1) = \sum_{i \in S} p_i - f|S| \geqslant opt(I) - 2fn$ since $|S| \leqslant 2n$.

Moreover $opt(I) \geqslant \max_{1 \leqslant i \leqslant n} p_i \geqslant (\sum_{i=1}^{n} p_i)/n = P/n$. Thus, considering $f = \frac{\varepsilon P}{2n^2}$, the value of the solution S' in I is $val(S') \geqslant opt(I) - \varepsilon\, opt(I)$.

The running time of the algorithm is $O(n(P/f)^2)$ that is $O(n^5/\varepsilon^2)$.

This bound is improved to $O(nP/f)$, that is $O(n^3/\varepsilon)$, when the network of the problem instance is a path. □

Corollary 13. OP *has an FPTAS on cycles with service time. For any given* $\varepsilon > 0$, *there is a* $(1-\varepsilon)$*-approximation algorithm with running time* $O(n^4/\varepsilon)$.

Proof. When the network is a cycle the error bound is obtained by a procedure like the one described in Theorem 11 using the FPTAS described in Theorem 12 to approximate $n+1$ problems within $1-\varepsilon$ and taking the best overall solution. □

4.2. Prize collecting TSP

Theorem 14. PCTSP *can be solved on trees with service time within the following pseudo-polynomial time bounds:*

(*i*) $O(nP_{\min}^2)$.
(*ii*) $O(nT^2)$, *where* $T = \sum t_{i,j}$.
(*iii*) $O(nP_{\min})$ *if the input is a path.*
(*iv*) $O(nT)$ *if the input is a path.*

Proof. Similarly to Theorem 10, the first step of the proposed algorithms consists in applying the two-step tree reduction so that we have to deal with a tree of at most $2n$ customers without service times.

Proof of case (*i*). For each vertex $i \geqslant 0$ we compute function values $z_i(p)$, that are computed traversing the tree in post-order, representing the minimum traveling time that must be spent in the subtree rooted in i in order to collect a profit at least p. Function values $z_i(p)$ being 0 when $p \leqslant 0$ are stored in a vector with $P_{\min}$ components for $p = 1, \ldots, P_{\min}$. Vector components are computed as (p_i and d_i are the profit and the number of children for vertex i, respectively)

$$z_i(p) = \begin{cases} 0 & \text{if } p \leqslant p_i; \\ +\infty & \text{if } d_i = 0; \\ 2t_{i,i_1} + z_{i_1}(p - p_i) & \text{if } d_i = 1; \\ \min \begin{cases} 2t_{i,i_1} + z_{i_1}(p - p_i) \\ 2t_{i,i_2} + z_{i_2}(p - p_i) \\ \min\limits_{1 \leqslant \ell \leqslant p - p_i - 1} (2(t_{i,i_1} + t_{i,i_2}) + z_{i_1}(\ell) + z_{i_2}(p - p_i - \ell)) \end{cases} & \text{if } d_i = 2. \end{cases}$$

The optimal value of the problem is $z_0(P_{\min})$. The optimal customer subset can be retrieved by storing for each $z_i(p)$ the children (if any) and their corresponding collected profits which have given positive contribution to the function value.

Postorder traversal takes $O(n)$ steps. At nodes i with $d_i < 2$ each component is computed in $O(1)$ time and the whole vector z_i is computed in $O(P_{\min})$ time. At nodes i with $d_i = 2$, $O(P_{\min})$ time is needed for each component and the whole vector z_i is computed in $O(P_{\min}^2)$ time. The maximum number of nodes with 2 children is $n/2$, thus all vectors z_i are computed in $O(nP_{\min}^2)$ time.

Proof of case (*ii*). For each vertex i we compute a function $z_i(t)$ for $t = 0, \ldots, T$ that can be stored in a vector with $T+1$ components. Here $z_i(t)$ represents the maximum profit that can be collected in time exactly t in the subtree rooted in the node i.

We consider $z_i(t) = -\infty$ if $t < 0$. Vector components are computed as

$$z_i(t) = \begin{cases} -\infty & \text{if } d_i = 0 \text{ and } t \neq 0; \\ p_i & \text{if } d_i = 0 \text{ and } t = 0; \\ p_i + z_{i_1}(t - 2t_{i,i_1}) & \text{if } d_i = 1; \\ p_i + \max \begin{cases} z_{i_1}(t - 2t_{i,i_1}) \\ z_{i_2}(t - 2t_{i,i_2}) \\ \max\limits_{2t_{i,i_1} \leqslant \ell \leqslant t - 2t_{i,i_2}} (z_{i_1}(\ell - 2t_{i,i_1}) + z_{i_2}(t - \ell - 2t_{i,i_2})) \end{cases} & \text{if } d_i = 2. \end{cases}$$

The optimum value corresponds to the minimum value of t such that $z_0(t) \geqslant P_{\min}$. The running time of the algorithm is $O(nT^2)$.

Proof of case (*iii*) and (*iv*). These parts are analogous to the proofs of (*iii*) and (*iv*) of Theorem 10, the difference is that we now substitute the optimization in the recursive step by

$$2(t_{i,i_1} + t_{i,i_2}) + z_{i_2}(p - p_i - p_{i_1})$$

and

$$p_{i_1} + z_{i_2}\left([t - 2t_{i,i_1}] - 2t_{i,i_2}\right)$$

respectively. □

Theorem 15. PCTSP *can be solved on cycles with service times within pseudo-polynomial time* $O(n^2 P_{\min})$ *and* $O(n^2 T)$ *where* $T = \sum t_{i,j}$.

Proof. Similarly to the notation used in the proof of Theorem 8, we assume that the indexing along the cycle is $0, 1, 2, \ldots, n, 0$ in this order, and every tour starts from, and returns to, 0. There can occur two essentially different types of a tour:

- Type 1 — all around the cycle;
- Type 2 — passing through vertices $1, \ldots, i$ in one direction, returning to 0, then in the other direction passing through $n, n-1, \ldots, j+1$, $j > i$ and returning to 0 via n.
  Note that in this case we do not assume that passing through a customer vertex implies a visit to the customer spending the corresponding service time.

Let us define Z_{-1} as the optimal solution value that can be obtained by a Type 1 tour and define Z_i $(i = 0, 1, \ldots, n)$ as the optimal solution value that can be obtained by a Type 2 tour reaching customer i as the farthest customer on the first round.

Evaluating Z_{-1} is equivalent to solve a Min Knapsack problem where item values are given by customer profits p_i, item weights are given by service times s_i and minimum weight is given by $P_{\min}$. This can be done by dynamic programming in $O(nP_{\min})$ and $O(nT)$.

Evaluating Z_i $(i = 0, \ldots, n)$ is equivalent to solve a path instance where the cycle network is transformed as follows. Edge $(i, i+1)$ is removed. Customers $j < i$ are relocated to the location of customer i. Customer n is detached from the depot and reconnected to customer i so that the new edge has traveling time equal to the original edge $(n, 0)$. According to Theorem 14, computing Z_i takes at most $O(nP_{\min})$ and $O(nT)$.

The problem can be solved taking the maximum over Z_h for $h = -1, 0, 1, \ldots, n$, that is in $O(n^2 P_{\min})$ and $O(n^2 T)$. □

Theorem 16. PCTSP *has an FPTAS on trees with service times. For any given* $\varepsilon > 0$, *there is a* $(1+\varepsilon)$*-approximation algorithm with running time* $O(n^3 (\ln T)^2/\varepsilon^2)$ *on trees and* $O(n^2 (\ln T)/\varepsilon)$ *on paths.*

Proof. We cannot apply directly the scaling method as in Theorem 12 since we have no bound on the optimum value for PCTSP. Thus, we proceed as follows. The time space between 0 and $T = \sum t_{i,j}$ is divided into $q+1$ intervals $[0, 1)$, $[1, (1+\varepsilon)^{1/n})$, $[(1+\varepsilon)^{1/n}, (1+\varepsilon)^{2/n})$, $\ldots$, $[(1+\varepsilon)^{q-1/n}, (1+\varepsilon)^{q/n})$ with $q = \lceil n \log_{1+\varepsilon} T \rceil$. In order to achieve an FPTAS, we adapt the dynamic programming algorithm from Theorem 14(*ii*) considering function z only for 0 and the q upper endpoints of the half-open intervals. More precisely an entry $z'_i(t') = p$, where t' is an upper endpoint of an interval of the traveling time space, indicates that there exists a tree rooted in i with profit p and traveling time *at most* t'.

We update z' bottom-up in the tree, starting with the leaves (that are vertices of depth 0). The update operation when we look at a vertex i with one child i_1 is as follows:

$$z'_i(t') = p_i + z'_{i_1}(v'_1)$$

where v'_1 is such that $t'/(1+\varepsilon)^{1/n} \leqslant v'_1 + 2t_{i,i_1} \leqslant t'$. When i has 2 children i_1, i_2, then

$$z'_i(t') = p_i + \max\left\{ z'_{i_1}(v'_1), z'_{i_2}(v'_2), \max_{2t_{i,i_1} \leqslant \ell \leqslant t - 2t_{i,i_1}} \left\{ z'_{i_1}(u'_1) + z'_{i_2}(u'_2) \right\} \right\}$$

where v'_1, v'_2, u'_1, u'_2 are such that $t'/(1+\varepsilon)^{1/n} \leqslant v'_1 + 2t_{i,i_1} \leqslant t'$, $t'/(1+\varepsilon)^{1/n} \leqslant v'_2 + 2t_{i,i_2} \leqslant t'$, $u'_1/(1+\varepsilon)^{1/n} \leqslant \ell - 2t_{i,i_1} \leqslant u'_1$, $u'_2/(1+\varepsilon)^{1/n} \leqslant t' - \ell - 2t_{i,i_2} \leqslant u'_2$. Moreover $z'_i(0) = p_i$ and $z'_i(t') = -\infty$ for a leaf i.

We prove in the following by induction that after performing all update operations, for any entry $z_i(t)$, $t = 0, \ldots, T$ from Theorem 14(*ii*), where i is a vertex of depth r, there exists an entry $z'_i(t')$ such that $z'_i(t') \geqslant z_i(t)$ and $t' \leqslant t/(1+\varepsilon)^{r/n}$. Thus, a $(1+\varepsilon)$ approximate value corresponds to a minimum t' such that $z'_0(t') \geqslant P_{\min}$.

The claim is clearly satisfied for vertices of depth 0. Assuming that it is true for $r-1$, we prove it for r. We distinguish between cases according to the recursion above.

Consider first t and a vertex i of depth r with one child i_1 of depth $r-1$. By induction there exist u'_1 such that $z'_{i_1}(u'_1) \geqslant z_{i_1}(t - 2t_{i,i_1})$ and $u'_1 \leqslant (t - 2t_{i,i_1})(1+\varepsilon)^{(r-1)/n}$. Let t' be the upper endpoint of the interval that contains the value $u'_1 + 2t_{i,i_1}$.

Then $t' \leqslant (u'_1 + 2t_{i,i_1})(1+\varepsilon)^{1/n} \leqslant (t - 2t_{i,i_1})(1+\varepsilon)^{r/n} + 2t_{i,i_1}(1+\varepsilon)^{1/n} \leqslant t(1+\varepsilon)^{r/n}$. Moreover $z'_i(t') = z'_{i_1}(v'_1) + p_i$ from the recursive computation. Since $v'_1 \geqslant u'_1$ by construction, we have $z'_{i_1}(v'_1) \geqslant z'_{i_1}(u'_1)$. Thus $z'_i(t') \geqslant z'_{i_1}(u'_1) + p_i \geqslant z_{i_1}(t - 2t_{i,i_1}) + p_i$, so $z'_i(t') \geqslant z_i(t)$.

If i of depth r has 2 children i_1, i_2, then a value $z_i(t)$ may originate from three different branches of the recursion. The first case using just the function z_{i_1} requires a word-by-word repetition of the argument above, and the second case using z_{i_2} follows analogously. Finally, when $z_i(t)$ is obtained from the combination of z_{i_1} and z_{i_2}, we observe that both i_1, i_2 have

depths at most $r-1$ and hence by the induction hypothesis both z_{i_1}, z_{i_2} admit $(1+\varepsilon)^{(r-1)/n}$ approximate values of z'_{i_1}, z'_{i_2} at the upper endpoints of the corresponding time intervals. Consequently, the sum of approximations properly approximates the sum for z_i.

The running time of this algorithm is $O(nq^2) = O(n^3(\ln T)^2/\varepsilon^2)$, which is polynomial in the input size and in $1/\varepsilon$.

If the network is a path, the time bound is $O(nq) = O(n^2(\ln T)/\varepsilon)$. □

Corollary 17. PCTSP *has an FPTAS on cycles with service times. For any given* $\varepsilon > 0$, *there is a* $(1+\varepsilon)$*-approximation algorithm with running time* $O(n^3(\ln T)/\varepsilon)$.

Proof. When the network is a cycle the error bound is obtained by a procedure like the one described in Theorem 15 using the FPTAS described in Theorem 16 to approximate $n+1$ problems within $1+\varepsilon$ and taking the best overall solution. □

Acknowledgements

We are grateful to two anonymous referees for their comments and suggestions that helped us to improve a previous version of the paper.

References

[1] A. Archer, M. Bateni, M. Hajiaghayi, H. Karloff, Improved approximation algorithms for prize-collecting Steiner tree and TSP, SIAM J. Comput. 40 (2011) 309–332.
[2] E.M. Arkin, J.S.B. Mitchell, G. Narasimhan, Resource-constrained geometric network optimization, in: Proceedings of the 14th Annual Symposium on Computational Geometry, 1998, pp. 307–316.
[3] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, J. ACM 45 (1998) 753–782.
[4] S. Arora, G. Karakostas, A $2+\varepsilon$ approximation algorithm for the k-MST problem, Math. Program. Ser. A 107 (2006) 491–504.
[5] B. Awerbuch, Y. Azar, A. Blum, S. Vempala, New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen, SIAM J. Comput. 28 (1999) 254–262.
[6] N. Bansal, A. Blum, S. Chawla, A. Meyerson, Approximation algorithms for deadline-TSP and vehicle routing with time-windows, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 166–174.
[7] D. Bienstock, M. Goemans, D. Simchi-Levi, D. Williamson, A note on the prize collecting traveling salesman problem, Math. Program. 59 (1993) 413–420.
[8] A. Blum, S. Chawla, D.R. Karger, T. Lane, A. Meyerson, M. Minkoff, Approximation algorithms for orienteering and discounted-reward TSP, SIAM J. Comput. 37 (2007) 653–670.
[9] A. Blum, R. Ravi, S. Vempala, A constant-factor approximation algorithm for the k-MST problem, J. Comput. System Sci. 58 (1999) 101–108.
[10] C. Chekuri, A. Ene, N. Korula, Prize-collecting Steiner tree and forest in planar graphs, eprint, arXiv:1006.4357v1 [cs.DS], 22 June 2010.
[11] C. Chekuri, N. Korula, M. Pál, Improved algorithms for orienteering and related problems, ACM Trans. Algorithms 8 (July 2012), 27 pages, Article 23.
[12] A.M. Costa, J.-F. Cordeau, G. Laporte, Steiner tree problems with profits, INFOR Inf. Syst. Oper. Res. 44 (2006) 99–116.
[13] D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problems with profits, Transp. Sci. 39 (2005) 188–205.
[14] P. Feofiloff, C.G. Fernandes, C.E. Ferreira, J. Coelho de Pina, Primal–dual approximation algorithms for the prize-collecting Steiner tree problem, Inform. Process. Lett. 103 (2007) 195–202.
[15] N. Garg, A 3-approximation for the minimum tree spanning k vertices, in: Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 302–309.
[16] N. Garg, Saving an epsilon: a 2-approximation for the k-MST problem in graphs, in: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2005, pp. 396–402.
[17] M.X. Goemans, D.P. Williamson, A general approximation technique for constrained forest problems, SIAM J. Comput. 24 (1995) 296–317.
[18] D.S. Johnson, M. Minkoff, S. Phillips, The prize collecting Steiner tree problem: theory and practice, in: Proceedings of the 11th ACM–SIAM Symposium on Discrete Algorithms, San Francisco, 2000, pp. 760–769.
[19] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.
[20] G.W. Klau, I. Ljubic, P. Mutzel, U. Pferschy, R. Weiskircher, The fractional prize-collecting steiner tree problem on trees, in: Proceedings of the 11th Annual European Symposium on Algorithms, ESA 2003, in: LNCS, vol. 2832, 2003, pp. 691–702.