ELSEVIER

Discrete Applied Mathematics 75 (1997) 201–216

DISCRETE APPLIED MATHEMATICS

$(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective

Igor Averbakh,[a,*] Oded Berman[b]

[a] *Mathematics Department, Western Washington University, Bellingham, WA-98225-9063, USA*
[b] *Faculty of Management, University of Toronto, Toronto, Ont., Canada M5S 1V4*

Received 17 February 1995; revised 17 June 1996

Abstract

Suppose p traveling salesmen must visit together all points/nodes of a tree, and the objective is to minimize the maximum of lengths of their tours. For location–allocation problems (where both optimal home locations of the salesmen and their tours must be found), which are NP-complete, fast polynomial heuristics with worst-case relative error $(p-1)/(p+1)$ are presented

Keywords: Traveling salesman; Approximate algorithms; Complexity

1. Introduction

We consider minmax p-traveling salesmen problems (p-TSP) on a tree, that can be interpreted as follows. There are p identical service units (servers), initially situated at some points of the tree (home locations). They are required to visit (serve) some set DP of demand points (each point from DP must be visited by at least one server) and return back to their home locations; DP is either the set of all nodes or the set of all points of the tree. The objective is to minimize the maximum of lengths of their tours. For the sake of convenience this interpretation will be used throughout the paper.

Problems of this type arise in many services such as repair and maintenance, delivery and customer pick-up. The minmax objective may be motivated, first, by the desire to distribute the workload to the servers in a "fair" way, second, by natural restrictions such as limited working day of the servers. The minmax p-TSP on a general network with a priori given single home location for all servers (depot) was

* Corresponding author. E-mail: averbakh@lake.scar.utoronto.ca.

0166-218X/97/$17.00
PII S0166-218X(96)00096-0

studied in [3, 2]. Franka et al. [2] discussed tabu search heuristics as well as an exact algorithm based on the solution of a closely related Distance Constrained Vehicle Routing Problem [5]. These approaches, although do not have any theoretical worst-case guarantees, perform satisfactorily in practice; Franka et al. [2] report solving to optimality problems involving up to 50 nodes. Frederickson et al. [3] studied worst-case behavior of several heuristics; in particular, they suggested a tour-partitioning heuristic which has a worst-case relative error of $e + 1 - 1/p$, where e is the worst-case error for the corresponding single-server algorithm (which is used as a subroutine). Thus, if the single-server TSP can be solved to optimality, the tour-partioning heuristic of [3] for the minmax p-TSP with a single given depot has a relative error not greater than $1 - 1/p$.

Averbakh and Berman [1] studied the minmax 2-TSP (two servers) with given home locations on a tree. For this (NP-complete) problem they presented a linear-time heuristic with worst-case relative error $\frac{1}{3}$ for the case of equal home locations and $\frac{1}{2}$ for the case of different home locations.

In this paper, we study the location–allocation (or routing-location) version of the minmax p-TSP on a tree, where home locations of servers are not given in advance and should be chosen along with servers' tours. This version of the problem is less restrictive, since the assumption of given locations clearly reduces flexibility in choosing tours. For this NP-complete problem, we develop fast heuristics with a worst-case relative error of $(p - 1)/(p + 1)$.

2. Problem formulation and notation

Consider a tree $T = (V, E)$ with V the set of nodes and E the set of (undirected) edges, $|V| = n$. T will also denote the set of all points of T. Throughout the paper, the term "subtree" is used in the topological sense, i.e. T' is a subtree of T iff T' is a connected subset of T (not necessarily closed). For any subtree $T' \subset T$, let $L(T')$ denote the total length of T'; L denotes the total length of T (the sum of the lengths of all the edges).

The subtree visited by a server in his service tour is referred to as allocation for that server.

The TSP for the case of a single server on a tree is trivial: it is well-known that any depth-first tour solves the problem, and the length of the optimal tour is equal to twice the total length of the tree. Without loss of generality, we assume that the service tour of each server is a depth-first tour within his allocation with length equal to twice the total length of the allocation. Due to this assumption, the considered problems can be formulated in graph-theoretic terms, and we use total lengths of allocations instead of lengths of service tours.

We consider location–allocation problems, where it is required to find optimal home locations and the corresponding optimal allocations given a set HL of possible home locations and the set DP of demand points.

Problem 1 (*Location–allocation minmax p-TSP*). Given an integer $p \geq 2$, sets $HL \subset T$ and $DP \subset T$, find locations $a_1, \ldots, a_p \in HL$ (not necessarily different) and closed subtrees $F_1, \ldots, F_p \subset T$ (allocations) such that $DP \subset F_1 \cup \cdots \cup F_p$ and $a_i \in F_i$, $i = 1, \ldots, p$, so as to minimize $\max\{L(F_1), \ldots, L(F_p)\}$.

We distinguish between four variants of Problem 1 with notation "Problem 1-X_1/X_2", where $X_1 = V$ if $HL = V$ and $X_1 = E$ if $HL = T$, $X_2 = V$ if $DP = V$ and $X_2 = E$ if $DP = T$. For example, if $HL = V$, $DP = T$, then Problem 1 is referred to as Problem 1-V/E. When a variant of the problem is not specified, reported results pertain to all the four variants.

We use the following notation and definitions. For any two points $a, b \in T$, let $d(a, b)$ denote the distance between a and b. For an edge (v_1, v_2) let $x(v_1, v_2; r)$ denote the point of edge (v_1, v_2) which is r units away from $v_1 (0 < r < d(v_1, v_2))$, $x(v_1, v_2; 0) = v_1$, $x(v_1, v_2; d(v_1, v_2)) = v_2$. For any constant $\gamma > 0$ a finite set $\Gamma \subset T$ is referred to as "γ-dividing set", if after deleting all the points of Γ from T tree T will be divided into connected components of lengths not greater than γ. A γ-dividing set Γ is called a minimum γ-dividing set, if it contains the minimum number of points (among all γ-dividing sets). For any node v, connected components of set $T \setminus \{v\}$ are referred to as v-branches, and for another point c, let $B(v, c)$ denote the unique v-branch that contains c (notice that $v \notin B(v, c)$). For any edge (a, b) it is assumed that points a, b do not belong to the edge; $[a, b]$ denotes edge (a, b) with its end points a, b. For any two points c, d of the tree, let $P(c, d)$ denote the path between c and d. $L^*_{E/E}$ denotes the optimal objective value for Problem 1-E/E; $L^*_{E/V}$, $L^*_{V/E}$, $L^*_{V/V}$ are defined analogously.

3. An auxiliary problem

We start with the following auxiliary problem.

Problem 2 (*Minimum γ-dividing set problem*). Given $\gamma > 0$, find a minimum γ-dividing set Γ for tree T.

Consider an algorithm for solving Problem 2.

Algorithm 1. We will use an auxiliary tree $\hat{T} = (\hat{V}, \hat{E})$, which will be changed during the action of the algorithm (before each step, current tree $\hat{T}$ represents the part of T that has not been examined yet). At the beginning $\hat{V} = V$, $\hat{E} = E$, $\Gamma = \emptyset$. At every step of the algorithm nodes $v \in \hat{V}$ of tree $\hat{T}$ have labels $s(v)$. Initially, all the labels are equal to 0.

Step k, $k = 1, 2, \ldots$. *Case* 1. Current tree $\hat{T}$ has more than one node. Take any end node w of current tree $\hat{T}$. Let v be the node of current tree $\hat{T}$ adjacent to node w.

Substep 1. If $s(w) \geqslant \gamma$, then put node w into set Γ. Change label $s(w)$ as follows: $s(w) = 0$ and go to substep 2.

If $s(w) < \gamma$, go to substep 2.

Substep 2. If $s(w) + d(w, v) \leqslant \gamma$, then delete edge (w, v) with node w from tree $\hat{T}$ and change label $s(v)$ as follows: $s(v) := s(v) + s(w) + d(w, v)$. Go to step $k + 1$.

If $s(w) + d(w, v) > \gamma$, then put points $x(w, v; \gamma - s(w))$, $x(w, v; 2\gamma - s(w))$, ..., $x(w, v; (\lceil (d(w, v) + s(w))/\gamma \rceil - 1)\gamma - s(w))$ consecutively into Γ (in total $\lceil (d(w, v) + s(w))/\gamma \rceil - 1$ points). Delete edge (w, v) with node w from $\hat{T}$ and change label $s(v)$ as follows:

$$s(v) := s(v) + d(w, v) + s(w) - \left(\left\lceil \frac{d(w, v) + s(w)}{\gamma} \right\rceil - 1 \right) \gamma.$$

Go to step $k + 1$.

Case 2: Current tree $\hat{T}$ is a node v.

If $s(v) > \gamma$, put node v into Γ and STOP.

If $s(v) \leqslant \gamma$, then STOP.

The description of the algorithm is completed.

Lemma 1. *The number of steps of Algorithm* 1 *is n.*

Proof. Trivial since at each step except the last step exactly one edge from $\hat{T}$ is deleted.

Lemma 2. (a) $|\Gamma| \leqslant \lceil L/\gamma \rceil - 1$;

(b) Γ *is a minimum* γ*-dividing set for* T.

Proof. For each point τ included in Γ by Algorithm 1 we define a corresponding subtree $Q(\tau)$ which will play an important role in further constructions. To define subtrees $Q(\tau)$, $\tau \in \Gamma$, consider a new auxiliary tree $\tilde{T}$, which is initially equal to T and is changed during the action of Algorithm 1. Let $\tau_1, \tau_2, \ldots, \tau_{|\Gamma|}$ be the points from set Γ obtained by Algorithm 1, ordered according to their appearing in Γ during the action of Algorithm 1. We delete subtree $Q(\tau)$ from tree $\tilde{T}$ as soon as point τ is put into Γ by Algorithm 1. This subtree is defined as follows. Suppose point τ is put into Γ at step k of Algorithm 1. If τ is the last point $\tau_{|\Gamma|}$ from Γ, then $Q(\tau)$ is the whole current tree $\tilde{T}$ (at the instant when τ is put into Γ). If $\tau \neq \tau_{|\Gamma|}$, then removing the point τ from current tree $\tilde{T}$ (at the instant when τ is put into Γ) divides $\tilde{T}$ into several connected components, and only one of these components has length greater than γ. Let it be component Ψ; we define $Q(\tau)$ as $\tilde{T} \setminus \Psi$. Notice that some of sets $Q(\tau)$ can be not closed.

An example of action of Algorithm 1 and the corresponding sets $Q(\tau)$, $\tau \in \Gamma$ are demonstrated in Appendix A.

Sets $Q(\tau)$, $\tau \in \Gamma$ have the following properties:

Property 1. $\tau \in Q(\tau)$ for all $\tau \in \Gamma$, and τ is the only point from Γ in $Q(\tau)$.

Property 2. $Q(\tau') \cap Q(\tau'') = \emptyset$, if $\tau' \neq \tau''$.

Property 3. Each set $Q(\tau)$ is a subtree of T with length not smaller than γ, and $Q(\tau_{|\Gamma|})$ has length strictly greater than γ.

Property 4. $Q(\tau_1) \cup Q(\tau_2) \cup \cdots \cup Q(\tau_{|\Gamma|}) = T$.

Property 5. Subtree $Q(\tau)$ has length $s(\tau)$, if τ was put into Γ at substep 1 or at the last step; $s(\tau)$ here is the label of τ just before putting τ into Γ. If τ was put into Γ at substep 2 and $\tau \neq \tau_{|\Gamma|}$, then $Q(\tau)$ has length γ.

Property 6. Removing point τ decomposes $Q(\tau)$ into connected components of lengths not greater than γ.

Property 7. If $\tau_{|\Gamma|}$ is an interior point of an edge ($\tau_{|\Gamma|}$ is the last point included in Γ), then $Q(\tau_{|\Gamma|})$ contains a node, and $L(Q(\tau_{|\Gamma|})) \leqslant 2\gamma$.

Property 8. For any $\tau \in \Gamma$ such that $\tau \neq \tau_{|\Gamma|}$ and τ is an interior point of an edge, $L(Q(\tau)) = \gamma$.

From Properties 1–4, we obtain the first statement of the lemma. From Property 6 and the definition of sets $Q(\tau)$, we obtain that Γ is a γ-dividing set. Now, it is easy to see that for any other γ-dividing set Γ', each one of sets $Q(\tau)$, $\tau \in \Gamma$ must contain at least one point from Γ'. Using Properties 1 and 2, we obtain the second statement of the lemma. □

Lemmas 1 and 2 imply

Theorem 1. *Algorithm* 1 *solves Problem* 2 *in time* $O(n + \lfloor L/\gamma \rfloor)$.

4. A $(p-1)/(p+1)$-heuristic for Problem 1-E/E

Using a reduction from the Multiprocessor Scheduling Problem [4] or from problem Partition [4], it is fairly easy to prove the following.

Theorem 2. *Problem* 1 *is NP-complete for each one of variants* V/V, V/E, E/V, E/E *for any fixed* $p \geqslant 2$. *If* p *is variable, the problem is strongly NP-complete. The results hold even for stars (trees where all edges have a common node).*

Thus, it would be interesting to obtain fast heuristics for solving the problem with good guarantees for the worst-case performance.

Notice that in some cases optimal allocations $F_i, i = 1, \ldots, p$ inevitably have intersections of non-zero lengths; for example, consider Problem 1 for the tree in Fig. 1, $p = 2$. For any optimal allocations F_1, F_2, edge (C, D) belongs to both of them.

Consider Heuristic H1 with running time complexity $O(\max\{n, p\})$ which for Problem 1-E/E obtains an approximate solution with value L^{H1} not greater than $2L/(p+1)$. Since $L^*_{E/E} \geqslant L/p$, the relative error $(L^{H1} - L^*_{E/E})/L^*_{E/E}$ for the heuristic's performance is not greater than $(p-1)/(p+1)$.

The idea of Heuristic H1 is as follows. First, using Algorithm 1, we obtain a minimum $L/(p+1)$-dividing set $\Gamma(|\Gamma| \leqslant p$ according to Lemma 2). The points from Γ divide tree T into subtrees of lengths not greater than $L/(p+1)$. Second, each one of these subtrees is assigned to one of the points from Γ incident to that subtree, so that each subree is assigned to exactly one point from Γ and the total length L_τ of subtrees assigned to any $\tau \in \Gamma$ is not smaller than $L/(p+1)$ (subtrees assigned to τ will be served by servers located at τ). Third, for any $\tau \in \Gamma$, $\max\{1, \lceil L_\tau/(L/(p+1))\rceil - 1\}$ servers are located at τ (there will be not more than p servers in total) and the subtrees assigned to τ are distributed to these servers in such a way that the allocation of each server has length not greater than $2L/(p+1)$.

Heuristic H1. (1) Algorithm 1 with $\gamma = L/(p+1)$ is applied to tree T and a minimum $L/(p+1)$-dividing set Γ with corresponding sets $Q(\tau), \tau \in \Gamma$ (see the proof of Lemma 2) is obtained (points of $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_{|\Gamma|}\}$ are ordered according to their appearance in Γ in the course of Algorithm 1).

(2) For each $\tau \in \Gamma$, $n(\tau) = \max\{1, \lceil((L(Q(\tau))/(L/(p+1))\rceil - 1\}$ servers are located at τ, where $L(Q(\tau))$ is the length of set $Q(\tau)$. The servers located at τ will serve together the set $Q(\tau)$ (with its boundary points).

(3) According to Property 6 of sets $Q(\tau)$ (see the proof of Lemma 2), set $Q(\tau)$ is a union of several branches (see Fig. 2) of lengths not greater than $L/(p+1)$ having

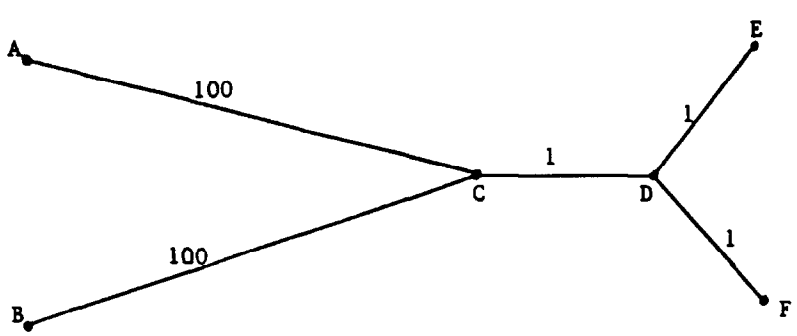

Fig. 1. For any optimal allocations $F_1, F_2 (p = 2)$ edge (C, D) belongs to both of them.

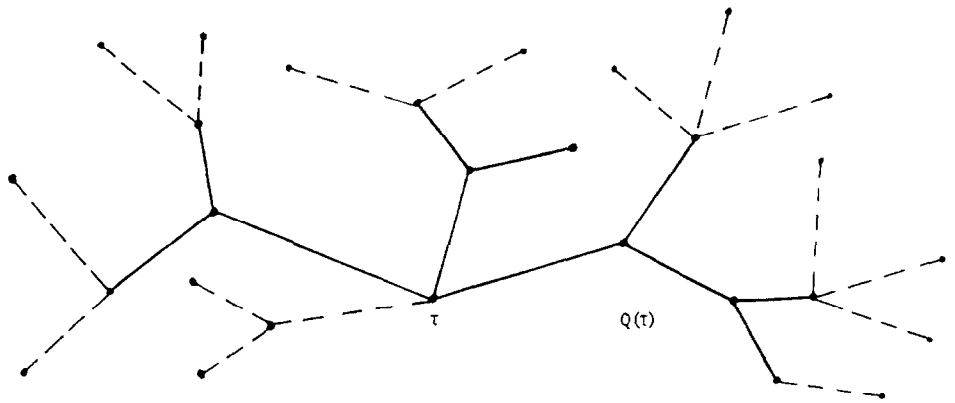

Fig. 2. τ and set $Q(\tau)$. $Q(\tau)$ (depicted with solid lines) is the union of three branches having common point τ.

the only common point τ. The total length of these branches is equal to $L(Q(\tau))$ and is not greater than $(n(\tau)+1)L/(p+1)$. These branches (with their boundary points) are assigned to $n(\tau)$ servers situated at τ so that: (a) each branch is assigned to exactly one server; (b) allocation of each server has length not greater than $2L/(p+1)$. The branches are assigned in consecutive order; a branch is assigned to any server which can accept it without exceeding limit $2L/(p+1)$ for the length of his allocation.

The description of Heuristic H1 is completed.

Illustrating examples for Heuristic H1 are provided in Appendix B.

Taking into account Property 3 of sets $Q(\tau)$ (see the proof of Lemma 2), we have

$$n(\tau)\frac{L}{p+1} \leqslant L(Q(\tau)) \leqslant (n(\tau)+1)\frac{L}{p+1}, \quad \tau \in \Gamma;$$

by summing up these inequalities for all $\tau \in \Gamma$ we obtain

$$\frac{L}{p+1}\sum_{\tau\in\Gamma} n(\tau) \leqslant L \leqslant \frac{L}{p+1}\sum_{\tau\in\Gamma}(n(\tau)+1)$$

and

$$\sum_{\tau\in\Gamma} n(\tau) \leqslant p+1 \leqslant \sum_{\tau\in\Gamma}(n(\tau)+1). \tag{1}$$

The first inequality in (1) must be strict because for the point $\tau_{|\Gamma|}$ (the last point included in Γ), $n(\tau_{|\Gamma|})L/(p+1) < L(Q(\tau_{|\Gamma|}))$ (since $L(Q(\tau_{|\Gamma|})) > L/(p+1)$, according to Property 3 of sets $Q(\tau)$). Therefore,

$$\sum_{\tau\in\Gamma} n(\tau) \leqslant p.$$

Thus, Heuristic H1 obtains an approximate solution for Problem 1-E/E. Since each server's allocation has length not greater than $2L/(p+1)$ and $L^*_{E/E} \geqslant L/p$, the relative

error for the heuristic performance is not greater than $(p-1)/(p+1)$. This bound is tight and cannot be improved; an example of tightness – a tree consisting of only one edge of length L (for this example $(L^{H1} - L^*_{E/E})/L^*_{E/E} = (p-1)/(p+1)$. Thus, the following theorem is proved.

Theorem 3. *Heuristic* H1 *finds an approximate solution to Problem* 1-E/E *in time* $O(\max\{n, p\})$ *with worst-case relative error* $(p-1)/(p+1)$. *The value of the obtained solution is not greater than* $2L/(p+1)$.

Corollary. $L^*_{E/E} \leqslant 2L/(p+1)$.

The bound in the Corollary from Theorem 3 is tight, as the example of a star tree with common node a and $p+1$ edges of equal lengths demonstrates. The length of each edge is equal to $L/(p+1)$, and at least one server must serve two edges.

Notice that Heuristic H1 can leave some servers idle (i.e. to use less than p servers). To be rigorous, the idle servers may be located at arbitrary nodes of the tree with allocations consisting of single nodes.

5. A $(p-1)/(p+1)$-heuristic for Problem 1-V/E

Consider Problem 1-V/E, where all servers have to be located at nodes but must serve together all points of tree T. Clearly, the optimal value $L^*_{V/E}$ for Problem 1-V/E is not less than $\max\{L/p, l_{\max}/2\}$, where $l_{\max}$ is the length of the longest edge of T. Below we present Heuristic H2, which finds an approximate solution to Problem 1-V/E with value L^{H2} not greater than $\max\{l_{\max}/2, 2L/(p+1)\}$; therefore, the relative error is not greater than $(p-1)/(p+1)$. We also show that this is the worst-case relative error.

First, notice that according to Property 7 of sets $Q(\tau)$ (see the proof of Lemma 2) and according to the description of Heuristic H1, if $\tau_{|\Gamma|}$ is an interior point of an edge, then only one server is located at $\tau_{|\Gamma|}$ and the allocation of that server contains a node. Therefore, we can relocate that server at a node without changing any allocations. From now on, we will refer to Heuristic H1 assuming that this minor modification is performed (it will be necessary for purely technical reasons, specifically, for Observation 1 below to be true).

The approximate solution to Problem 1-E/E obtained by Heuristic H1 has the following property.

Observation 1. If $k \geqslant 1$ servers are located inside some edge (c, d) by Heuristic H1 (i.e. at interior points of that edge), then the allocation of one of these servers contains a node (either c or d), and allocations of the other $k-1$ servers are subintervals of edge (c, d). Allocations of all the k servers have lengths equal to $L/(p+1)$, according to Property 8 from the proof of Lemma 2 (see Fig. 3).

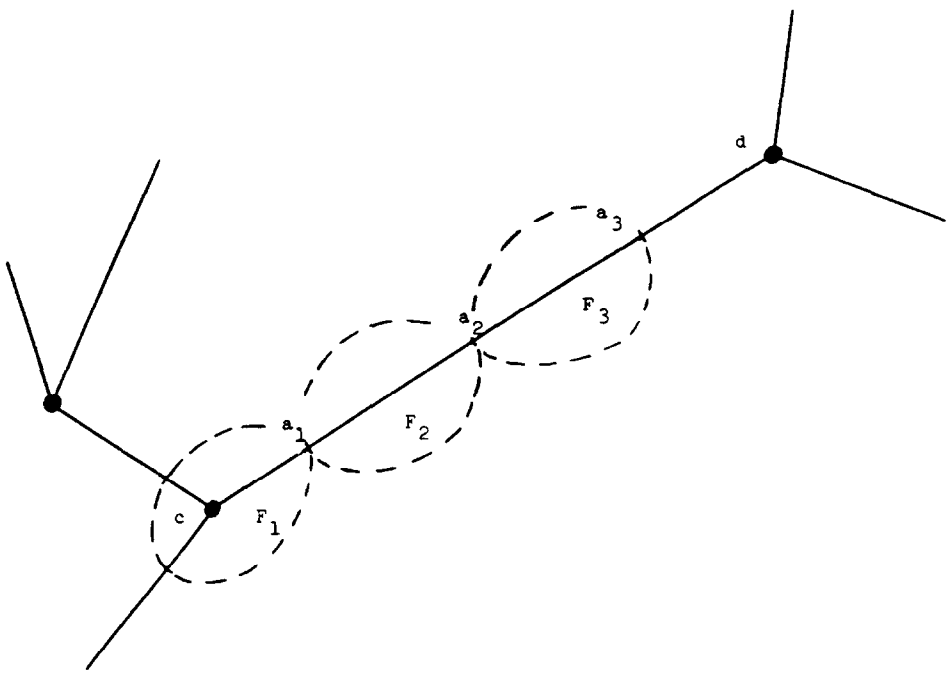

Fig. 3. F_1, F_2, F_3–allocations of the servers located at a_1, a_2, a_3, respectively; $L(F_1) = L(F_2) = L(F_3) = L/(p+1)$.

Using Observation 1, it is not difficult to transform the solution obtained by Heuristic H1 into an approximate solution to Problem 1-V/E with value not greater than $\max\{l_{\max}/2, 2L/(p+1)\}$ (and, thus, with relative error not greater than $(p-1)/(p+1)$). This is the main idea of Heuristic H2.

Heuristic H2.

Stage 1. Apply Heuristic H1 (with the modification mentioned above). Let $a_1, \ldots, a_p, F_1, \ldots, F_p$ be the obtained approximate solution to Problem 1-E/E.

Stage 2. If all locations $a_1, \ldots, a_p$ are nodal, output the solution obtained in Stage 1. Otherwise, for each edge (c, d) such that there are servers located inside (c, d), perform the following:

1. If there is only one server located inside (c, d), then, according to Observation 1, his allocation contains a node (say, c). Relocate the server at c without changing his allocation.

2. If there are exactly two servers located inside (c, d), then, according to Observation 1, allocation of one of the servers contains a node (say, c) and allocations of both servers have lengths equal to $L/(p+1)$. Moreover, it can easily be observed that both allocations are adjacent (have a common point).

Delete both servers; instead of them, locate one server at c, and assign to him allocations of both deleted servers. Thus, the length of the allocation of the new server is equal to $2L/(p+1)$, and the total number of servers has reduced by 1.

3. If there are $k > 2$ servers located inside (c, d), then, according to Observation 1, allocations of at least $k-1$ of them are subintervals of (c, d). Delete these $k-1$ servers; instead of them, locate one server at c and assign to him interval $[c, x(c, d; d(c, d)/2]$; locate one server at d and assign to him interval $[x(c, d; d(c, d)/2), d]$. The allocations of these two new servers have lengths not greater than $l_{\max}/2$, and the total number of servers has not increased.

The descsription of Heuristic H2 is completed.

If Heuristic H2 leaves some servers idle, they may be located at arbitrary nodes with allocations consisting of single points.

Illustrating examples for Heuristic H2 are provided in Appendix B.

Theorem 4. *Heuristic H2 finds an approximate solution for Problem* 1-V/E *in running time* $O(\max\{n, p\})$ *with worst-case relative error* $(p-1)/(p+1)$. *The value of the obtained solution is not greater than* $\max\{l_{\max}/2, 2L/(p+1)\}$.

Proof. As follows from the above discussion, the value L^{H2} of the approximate solution obtained by Heuristic H2 is not greater than $\max\{l_{\max}/2, 2L/(p+1)\}$, and the relative error is not greater than $(p-1)/(p+1)$. This bound on the relative error is achievable; an example of tightness–a path with $p+1$ edges of equal lengths (for this example $(L^{H2} - L^*_{V/E})/L^*_{V/E} = (p-1)/(p+1)$.
Thus, $(p-1)/(p+1)$ is the worst-case relative error. □

Corollary. $L^*_{V/E} \leqslant \max\{l_{\max}/2, 2L/(p+1)\}$.

Remark. It is not difficult to reduce the time complexity of Heuristic H2 to $O(n)$, but this does not seem to be an important improvement.

6. The case of nodal demand ($DP = V$).

Consider the Problem 1-V/V. Heuristic H2 can be modified to obtain an approximate solution to Problem 1-V/V with value not greater than $2L/(p+1)$. Notice that if F_t is one of the allocations obtained by Heuristic H2 and $L(F_t) > 2L/(p+1)$, then F_t is a subinterval of some edge $[v_1, v_2]$ of tree T such that $d(v_1, v_2) > 2L/(p+1)$, and each one of nodes v_1, v_2 is a home location for some server. Also, no edge of length greater than $2L/(p+1)$ is served entirely by a single server. Therefore, an approximate solution to Problem 1-V/V with value not greater than $2L/(p+1)$ can be obtained from the solution obtained by Heuristic H2 simply by deleting all interior points of all edges that are longer than $2L/(p+1)$ from all allocations. Modified in this way Heuristic H2 will be referred to as Heuristic H2′.

The upper bound $(p-1)/(p+1)$ for the relative error is not guaranteed for Heuristic H2′, because optimal value $L^*_{V/V}$ for Problem 1-V/V can be smaller than L/p. Nevertheless, using Heuristic H2′ it is possible to obtain an approximate solution to Problem 1-V/V with worst-case relative error $(p-1)/(p+1)$ in polynomial time, if p is fixed. We give a recursive description of the corresponding Algorithm 2-p.

Obviously for Problem 1-V/V there exist optimal allocations $F_1^*, \ldots, F_p^*$ such that either $F_1^* \cup \ldots \cup F_p^* = T$ or set $T \setminus (F_1^* \cup \ldots \cup F_p^*)$ contains at least one edge of T.

Algorithm 2-2 (case $p = 2$). Tree $T = (V, E)$ of length $L > 0$ is given. Let $l_1, \ldots, l_{n-1}$ be the edges of the tree. Any edge l_i divides tree T into two connected components $G_1(l_i)$, $G_2(l_i)$. Let $f(l_i) = \max\{L(G_1(l_i)), L(G_2(l_i))\}$.

Step 1. For all l_i, $i = 1, \ldots, n-1$ values $f(l_i)$ are calculated.

Step 2. Heuristic H2′ is applied to tree T, $p = 2$. Let $L^{H2'}$ be the value of allocations, F'_1, F'_2 obtained by Heuristic H2′ ($L^{H2'} = \max\{L(F'_1), L(F'_2)\}$).

Step 3. Calculate $f^* = \min\{f(l_1), \ldots, f(l_{n-1}), L^{H2'}\}$. If $f^* = L^{H2'}$, then take the solution obtained by Heuristic H2′ as an appoximate solution to Problem 1-V/V. If $f^* = f(l_i)$ for some i. take $F_1 = G_1(l_i)$, $F_2 = G_2(l_i)$ as an approximate solution to Problem 1-V/V (with some nodal home locations $a_1 \in F_1$, $a_2 \in F_2$).

Consider the general case of some fixed $p > 2$, assuming that Algorithm 2-t for $t = 2, \ldots, p-1$ is already defined.

Algorithm 2-p. Let $l_1, \ldots, l_{n-1}$ be the edges of tree T.

Step 1. For each l_i, $i = 1, \ldots, n-1$ the following procedure is applied. Let edge l_i divide tree T into two connected components $G_1(l_i)$ and $G_2(l_i)$. Apply Algorithm 2-t with $t = 1, \ldots, p-1$ to both $G_1(l_i)$ and $G_2(l_i)$ (we assume that Algorithm 2-t with $t = 1$ applied to any tree G simply gives that tree G and its length $L(G)$). Let $f_k^*(G)$ denote the value of allocations $F_1, \ldots, F_k$ obtained as a result of applying Algorithm 2-k to a tree G, $f_k^*(G) = \max\{L(F_1), \ldots, L(F_k)\}$. Let

$$f(l_i) = \min_{t \in \{1,2,\ldots,p-1\}} \{\max\{f_t^*(G_1(l_i)), f_{p-t}^*(G_2(l_i))\}\} \tag{2}$$

and let $t^*(l_i)$ be the minimizer in (2).

$$t^*(l_i) \in \operatorname*{Argmin}_{t \in \{1,2,\ldots,p-1\}} \{\max\{f_t^*(G_1(l_i)), f_{p-t}^*(G_2(l_i))\}\}.$$

Step 2. Heuristic H2′ is applied for tree T. Let $L^{H2'}$ be the value of allocations $F'_1, \ldots, F'_p$ obtained by Heuristic H2′, $L^{H2'} = \max\{L(F'_1), \ldots, L(F'_p)\}$.

Step 3. Calculate $f^* = \min\{f(l_1), \ldots, f(l_{n-1}), L^{H2'}\}$. If $f^* = L^{H2'}$, then take the solution obtained by Heuristic H2′ as an approximate solution for Problem 1-V/V. If $f^* = f(l_j)$ for some $j \in \{1, \ldots, n-1\}$, then take the allocations obtained by Algorithm $2 - t^*(l_j)$ applied to $G_1(l_j)$ and obtained by Algorithm $2 - (p - t^*(l_j))$ applied to $G_2(l_j)$ at step 1 as an approximate solution to Problem 1-V/V.

Theorem 5. *Algorithm* $2 - p (p \geqslant 2)$ *obtains an approximate solution to Problem* 1-V/V *with worst-case relative error* $(p-1)/(p+1)$. *The running time of the algorithm is* $O(n^{p-1})$ (p *is assumed to be fixed*).

Proof. The second statement of the theorem can easily be proved by induction (notice that step 1 of Algorithm 2-2 can be performed in $O(n)$ time using standard bottom-up dynamic programming). To prove the first statement, we start from the case $p = 2$ and then use induction on p. Let F_1^*, F_2^* be optimal allocations for Problem 1-V/V ($p = 2$) such that either $F_1^* \cup F_2^* = T$ or $T \setminus (F_1^* \cup F_2^*)$ is an edge of T. If $T \setminus (F_1^* \cup F_2^*)$ is an edge l_i, then $f(l_i)$ is calculated at step 1 and $f^* = f(l_i)$, i.e. Algorithm 2-2 obtains an optimal solution to Problem 1-V/V. If $T = F_1^* \cup F_2^*$, then optimal value $L_{V/V}^* = \max\{L(F_1^*), L(F_2^*)\}$ is not smaller than $L/2$, and f^* is smaller than $2L/3$ since Heuristic H2′ at $p = 2$ obtains a solution with value not greater than $2L/3$. Therefore, the relative error is not greater than $\frac{1}{3}$. This is also the worst-case relative error; an example of tightness – a star with six edges of equal lengths (in the worst case Heuristic H2′ assigns four edges to one server and two edges to the other server).

Now consider the general case $p = m > 2$, assuming that for $p < m$ the theorem is proved (this is the induction hypothesis). Let $F_1^*, \ldots, F_m^*$ be any optimal allocations for Problem 1-V/V such that either $F_1^* \cup \cdots \cup F_m^* = T$ or $T \setminus (F_1^* \cup \cdots \cup F_m^*)$ contains at least one edge l_i of T. In the latter case, according to the induction hypothesis

$$f(l_i) \leqslant (1 + (m-2)/m) \max\{L(F_1^*), \ldots, L(F_m^*)\},$$

because $f(l_i)$ is the value of allocations obtained either by Algorithm $2 - t^*(l_i)$ applied to $G_1(l_i)$ or by Algorithm 2-$(m - t^*(l_i))$ applied to $G_2(l_i)$ and $1 \leqslant t^*(l_i) < m$. Therefore, in this case Algorithm 2-m obtains an approximate solution with a relative error not greater than $(m-2)/m$.

Consider the other case where $F_1^* \cup \cdots \cup F_m^* = T$. Then the optimal value $L_{V/V}^* = \max\{L(F_1^*), \ldots, L(F_m^*)\}$ is not smaller than L/m, and f^* is not greater than $2L/(m+1)$ (since Heuristic H2′ obtains an approximate solution with value not greater than $2L/(m+1)$) and therefore the relative error is not greater than $(m-1)/(m+1)$. This is also the worst-case relative error; an example of tightness – a star with $m(m+1)$ edges of equal lengths. The theorem is proved. □

Remark. Notice that the case E/V needs no special consideration: any ε-optimal solution to Problem 1-V/V is also an ε-optimal solution to Problem 1-E/V, and therefore all results of this section can also be applied to the case E/V.

7. Conclusions and future research

In this paper, linear-time heuristics with worst-case relative error $(p-1)/(p+1)$ for (NP-complete) Problems 1-E/E and 1-V/E are obtained. An approximate algorithm with complexity $O(n)$ and worst-case relative error $(p-1)/(p+1)$ is developed for Problem 1-V/V.

As a possible direction for future research, it is interesting to try to find polynomial heuristics for Problem 1 with worst-case relative error smaller than $(p-1)/(p+1)$ (or to prove that such polynomial algorithms do not exist). Also, it would be interesting to answer the question: Is Problem 1 NP-complete in the strong sense for a fixed $p \geqslant 2$ or there exists a pseudopolynomial algorithm?

Acknowledgements

This research was supported by a grant from Natural Sciences and Engineering Research Council of Canada (NSERCC).

Appendix A

An illustrating example for Algorithm 1 and sets $Q()$. Consider the tree $T=(V, E)$ shown in Fig. 4, a; $L=17$, $=5$. At the beginning $\Gamma=\emptyset$, $s(A)=s(B)=s(C)=s(D)=s(\mathrm{E})=0$, $\hat{T}=T$. After the first step (node A is taken) tree $\hat{T}$ is shown in Fig. 4, b; $s(B)=1$, $s(C)=s(D)=s(E)=0$; $\Gamma=\{x(A, B; 5)\}$. After the second step (node D is taken) tree $\hat{T}$ is shown in Fig. 4, c; $s(B)=4$, $s(E)=s(C)=0$, $\Gamma=\{x(A, B; 5)\}$. After the third step (node E is taken) tree $\hat{T}$ is shown in Fig. 4 d; $s(B)=8$, $s(C)=0$, $\Gamma=\{x(A, B; 5)\}$. After the fourth step (node B is taken) tree $\hat{T}$ is shown in Fig. 4, e; $s(C)=4$, $\Gamma=\{x(A, B; 5), B\}$. After the fifth step $\Gamma=\{x(A, B; 5), B\}$. End.

$$Q(x(A, B; 5))=[A, x(A, B; 5)],$$

$$Q(B)=[x(A, B; 5), B][D, B][E, B][C, B]\{x(A, B; 5)\}.$$

Appendix B

Illustrating examples for Heuristics H1 and H2.

1. Consider Problem 1-E/E with $p=4$ for the tree in Fig. 5 ($L=20$) and apply to it Heuristic H1. Let Algorithm 1 with $=L/(p+1)=4$ take the nodes in the following order: a, b, c, d, e, f. Then $\Gamma=\{c, x(c, e; 4), x(c, e; 8), x(e, f; 1)\}$; $Q(c)=[a, c][b, c]$, $Q(x(c, e; 4))=[c, x(c, e; 4)]\{c\}$, $Q(x(c, e; 8))=[x(c, e; 4), x(c, e; 8)]\{x(c, e; 4)\}$, $Q(x(e, f; 1))=[x(c, e; 8), e][e, d][e, f]\{x(c, e; 8)\}$. Heuristic H1 locates one server at

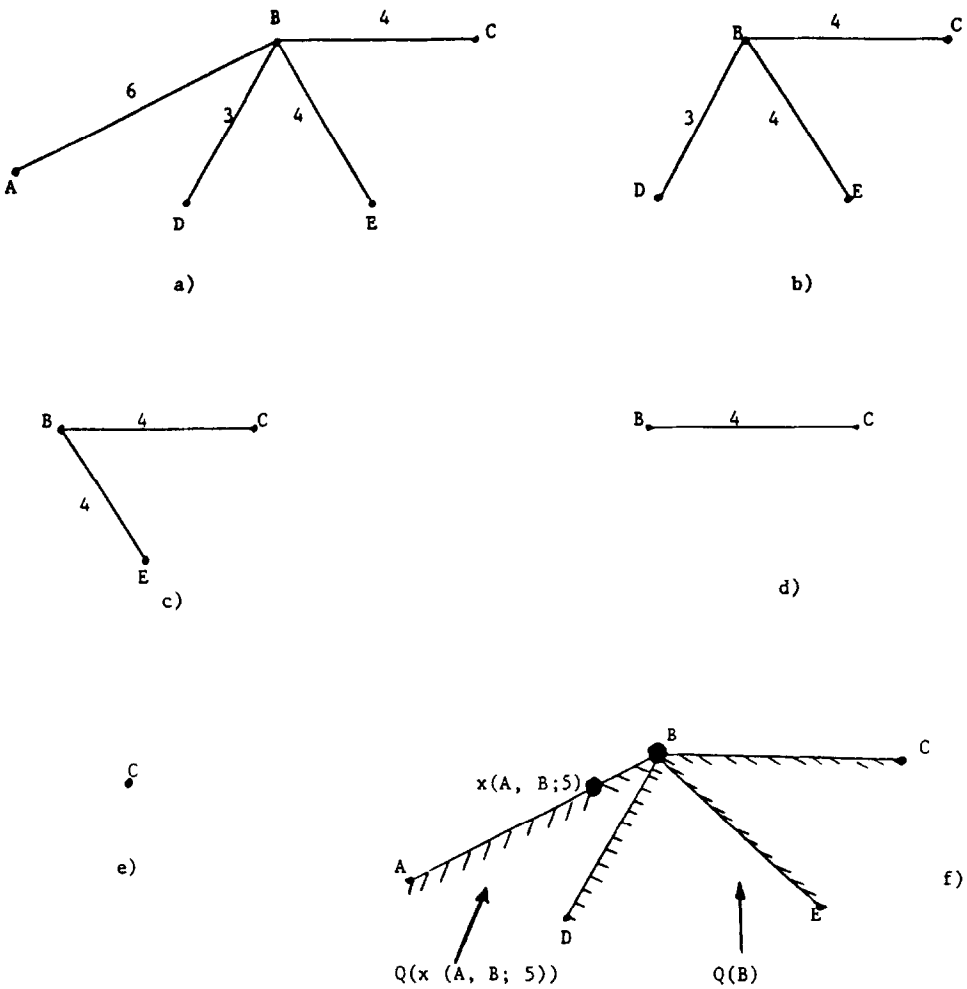

Fig. 4. Illustrating example for Algorithm 1 and sets $Q(\tau)$. $Q(x(A, B; 5)) = [A, x(A, B; 5)]$; $Q(B) = [x(A, B; 5), B] \cup [D, B] \cup [E, B] \cup [C, B] \setminus \{x(A, B; 5)\}$.

each one of the points of Γ, and the server located at $\tau \in \Gamma$ serves subtree $Q(\tau)$ (with its boundary points). The value of the obtained solution is $L(Q(x(e, f; 1))) = 8$. The optimal value is 5; the relative error is $(8 - 5)/5 = \frac{3}{5}$.

2. Consider the Problem 1-V/E with $p = 4$ for the same tree and apply to it Heuristic H2 (with the same order of taking nodes). Heuristic H2 locates two servers at node c and one server at node e. The first server located at c serves subtree $F_1 = [a, c] \cup [b, c]$; the second server located at c serves subtree $F_2 = [c, x(c, e; 8)]$; the third server located at e serves subtree $F_3 = [x(c, e; 8), e] \cup [e, d] \cup [e, f]$. The value of the solution is 8.

3. Consider the Problem 1-E/E with $p = 6$ for the tree demonstrated in Fig. 6 ($L = 28$) and apply to it Heuristic H1. Let Algorithm 1 with $\gamma = L/(p + 1) = 4$ take

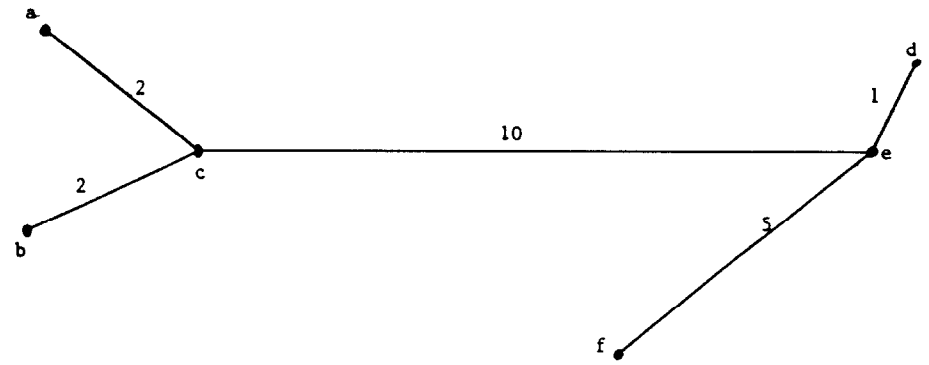

Fig. 5. Illustrating example for Heuristics H1 and H2.

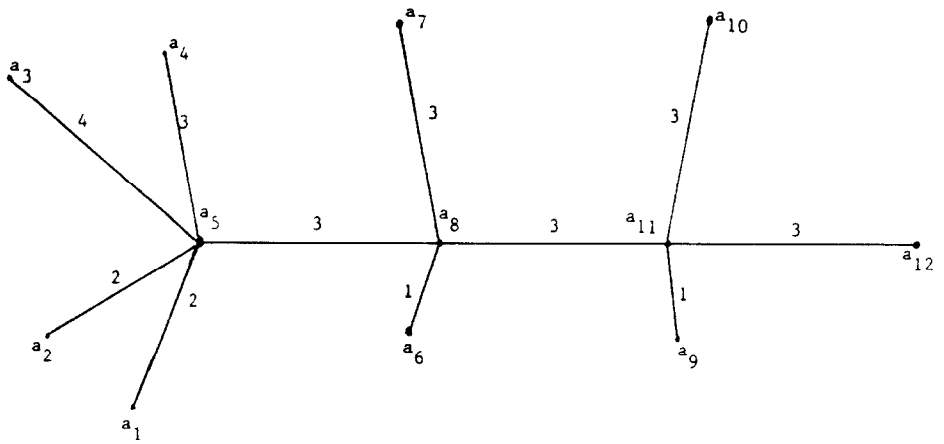

Fig. 6. Illustrating example for Heuristics H1 and H2.

the nodes in the following order: $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}$. Then $\Gamma = \{a_5, a_8, a_{11}\}$, $Q(a_5) = [a_1, a_5] \cup [a_2, a_5] \cup [a_3, a_5] \cup [a_4, a_5]$, $Q(a_8) = [a_5, a_8] \cup [a_6, a_8] \cup [a_7, a_8] \setminus \{a_5\}$, $Q(a_{11}) = [a_8, a_{11}] \cup [a_{10}, a_{11}] \cup [a_9, a_{11}] \cup [a_{12}, a_{11}] \setminus \{a_8\}$. Heuristic H1 locates two servers at a_5, one server at a_8 and two servers at a_{11}. The corresponding allocations are: $F_1 = [a_1, a_5] \cup [a_2, a_5] \cup [a_3, a_5]$, location at a_5; $F_2 = [a_4, a_5]$, location at a_5; $F_3 = [a_5, a_8] \cup [a_6, a_8] \cup [a_7, a_8]$, location at a_8; $F_4 = [a_8, a_{11}] \cup [a_9, a_{11}] \cup [a_{10}, a_{11}]$, location at a_{11}; $F_5 = [a_{11}, a_{12}]$, location at a_{11}. One server is left idle. The value of the obtained solution is 8. Heuristic H2 obtains the same solution.

References

[1] I. Averbakh and O. Berman, A heuristic with worst-case analysis for minmax routing of two traveling salesmen on a tree, Discrete Appl. Math., forthcoming.

[2] P.M. Franka, M. Gendreau, G. Laporte and F. Muller, The m-Travelling Salesman Problem with Minmax Objective. Centre de recherche sur les transports (Montreal), Publication #869 (1992).
[3] G.N. Frederickson, M.S. Hecht and C.E. Kim. Approximation algorithms for some routing problems, SIAM J. Comput. 7, (1978) 178–193.
[4] M. Garey and D. Johnson, Computers and Intractability, (Freeman, San Francisco, 1979).
[5] G. Laporte, M. Desrochers and Y. Nobert, Two exact algorithms for the distance-constrained vehicle routing problem, Networks 14 (1984) 161–172.