# The Multiple Traveling Salesman Problem on Spiders *

Pedro Pérez-Escalona[1], Ivan Rapaport[1,2], José Soto[1,2], and Ian Vidal[2]

[1] Departamento de Ingeniería Matemática, Universidad de Chile
{peperez,rapaport,jsoto}@dim.uchile.cl
[2] Centro de Modelamiento Matemático (UMI 2807 CNRS), Universidad de Chile
ian.vidal@ing.uchile.cl

**Abstract.** Given (i) a set of $N + 1$ vertices, that corresponds to $N$ *clients* and 1 *depot*, (ii) the travel time between each pair of vertices and (iii) a number $m$ of *salespersons*, the *multiple traveling salesman problem* consists in finding $m$ tours such that, starting from the depot, all clients are visited in such a way that some objective function is minimized. The objective function we consider in this paper is the *makespan*. More precisely, the goal is to find $m$ tours (one for each salesperson) that minimize the time elapsed from the beginning of the operation until the last salesman comes back to the depot. We take into account *handling times*, i.e., the time spent visiting each client, which we assume to be the same for all of them. We address the problem in the particular case where the *depot-clients network* is a spider, with the depot located at its center. We show that this case is NP-hard even for 2 salespersons. We also show structural properties of the optimal solutions. These properties allow us to devise a PTAS for minimizing the makespan. More precisely, a $(1 + \varepsilon)$-approximation algorithm with running time $N^{O(m/\varepsilon)}$.

**Keywords:** Multiple traveling salesman problem · Salesperson routing problem · Approximation algorithms · Polynomial-time approximation schemes

## 1 Introduction

The *Traveling Salesman Problem* (TSP) is probably the best-known problem in combinatorial optimization [12]. It finds countless applications in different fields, ranging from logistics [13] to electronic circuit manufacturing [8].

The *Multiple Traveling Salesman Problem* (mTSP [3]) is a natural generalization of the TSP. While in the TSP there is one salesman who must visit all the clients, in the mTSP there are $m$ salespersons, who must jointly visit all the clients. In turn, the Vehicle Routing Problem (VRP) is a generalization of the mTSP, where the role of the salespersons is played by (capacitated) vehicles [11].

* Additional support from ANID via FONDEF IDeA I+D ID18I10250, PIA/Apoyo a Centros Científicos y Tecnológicos de Excelencia AFB170001, Fondecyt 1170021 and Fondecyt 1181180.

In the mTSP there are $N$ clients, 1 depot, and $m$ salespersons that must visit all the clients starting from the depot. The standard objective function is to minimize the sum of the length of all $m$ tours. In this work we focus on the *min-max* version of the mTSP. More precisely, the goal is to find $m$ tours (one for each salesperson) that minimize the makespan: the time elapsed from the beginning of the operation until the last salesperson comes back to the depot. In the literature this problem is equivalent to the Single-Depot Min-Max Cycle Cover Problem (SDMMCCP). The aim of SDMMCCP is to cover all the vertices of a weighted graph with at most $m$ cycles, all of them passing through one particular vertex $D$, the depot, such that the weight of the heaviest cycle is minimized. Frederickson et al. devised a $\rho + 1$-approximation algorithms in [5], where $\rho$ is any approximation ratio for TSP (from Christofides [4] we know that $\rho \leq 3/2$).

The mTSP is NP-complete, as it is a generalization of the TSP. One way to tackle NP-complete problems is to study subproblems and variants. In graph problems a typical approach consists in restricting the topology of the network. For instance, in [14], the authors study the mTSP in trees. They devise, for fixed values of $m$, a pseudo-polynomial, $(1+\varepsilon)$-approximation algorithm running in time $O(N^{2m-1}/\varepsilon^{2k-2})$. Later, Becker and Paul develop, also for trees, a $(1+\varepsilon)$-approximation algorithm running in time $N^{O(\varepsilon^{-8})}$ [2].

In [16], Yu and Liu study the variant of mTSP with *release times*, which are times fixed by each client before which she/he cannot be visited. They study graphs as simple as paths, and they show that the problem of minimizing the makespan is polynomially solvable.

One can also take into account *handling times*, i.e., the time of visit to the clients. In fact, when each client has, together with his/her own release time, his/her own handling time, the problem is called *Multi-Vehicle Scheduling Problem* (VSP), and it is NP-complete even in a path [9, 10]. The authors also give, for every fixed $m$, a $(1+\varepsilon)$-approximation algorithm that runs in time $O((1+2/\varepsilon)m^2N^2(N+2^{1+2/\varepsilon})(mN^3(1+2/\varepsilon)/\varepsilon)^{m(1+2/\varepsilon))}$. On the other hand, Gaur et al. [7] give a polynomial-time $\frac{5}{3}$-approximation algorithm.

Bao and Liu study the VSP in cycles and trees [1]. They show, for cycles, a 12/7-approximation algorithm for minimizing the makespan. In the case of trees, they give a 9/5-approximation algorithm. In [15], Xu and Xu show a $\max\{3-2/m, 2\}$-aproximation for VSP in arbitrary graphs.

The topology we consider in the present paper is a spider. A spider is a tree with one distinguished vertex called the center and where every non-center vertex has degree at most 2. We study the mTSP when the depot is located at the center of a spider, and we consider uniform handling times (i.e, all clients have the same handling time). We denote this problem by mTSPHT for spiders.

In this paper we prove that mTSPHT in a spider is strongly NP-hard, even in a star (the complete bipartite graph $K_{1,N}$). So we cannot hope to obtain a fully polynomial time approximation scheme (FPTAS) unless P=NP. Our main contribution, together with establishing structural properties of the optimal solu-

tions, is to give a polynomial time approximation scheme (PTAS), more precisely, a $(1+\varepsilon)$-approximation algorithm running in time $N^{O(m/\varepsilon)}$.

Our approach is based on an interesting structural result, namely, the existence of particular type of solutions, that we call *well-structured*. Such solutions have the following very nice property: there is always one salesperson who visits just blocks of consecutive clients located at the end of some branches; once we remove such salesperson together with the clients he/she visited, the remaining (smaller) solution is also well-structured. From this result we can compute the solution of mTSPHT in the spider through dynamic programming.

As we have already mentioned, Becker and Paul develop, for trees, a $(1+\varepsilon)$-approximation running in time $N^{O(\varepsilon^{-8})}$ [2]. In the present work, the PTAS we develop is devised for spiders (particular types of trees) and runs in time $N^{O(m/\varepsilon)}$. The result of Becker and Paul is more general than ours (although they do not consider handling times). Nevertheless, we would like to discuss a little bit the difference when we restrict the two approaches to spiders.

Our algorithm, in the case of only one branch, takes time $O(N^2)$ and finds the optimal solution, while the one of [2] takes time $N^{O(\varepsilon^{-4})}$ and gives a $(1+\varepsilon)$-approximation. In the case of spiders with more branches, our algorithm gives a $(1+\varepsilon)$-approximation that takes time $N^{O(m/\varepsilon)}$, while the one of [2] takes time $N^{O(\varepsilon^{-8})}$. Note that, for fixed values of $m$, a situation which is already NP-hard for $m=2$, our algorithm takes time $N^{O(\varepsilon^{-1})}$.

## 2 Definitions and Basic Results

Let $G=(V\cup\{D\},E)$ be a simple, connected, undirected graph where $V$ is a set of $N$ clients and $D$ is a depot. The number of salespersons is $m$. The function $t\colon E\to\mathbb{R}_+$ represents the travel time between neighboring nodes. We consider the handling time, the time spent by any salesperson visiting each client, to be a constant $t_0\in\mathbb{R}_+$. We assume that the route taken by any salesperson between two nodes $u$ and $v$ is always the shortest one (in terms of time). Therefore, we will replace the function $t\colon E\to\mathbb{R}_+$ with its metric completion $t\colon \binom{V}{2}\to\mathbb{R}_+$ so that the travel time between any two nodes is well-defined. We reserve the term *distance* between two nodes to refer to the minimum number of edges in a path from one node to the other.

For any $r\in\mathbb{N}_+$, we define a *route* $S$ as a sequence of $r+1$ nodes in $V\cup\{D\}$, $(s_j)_{j=0}^r$, where $s_0=D$. Nodes $s_1$ and $s_r$ are, respectively, the first and last clients visited in $S$. We note by $C(S)$ the set of $r$ clients visited in the route $S$ (observe that $C(S)$ may be a strict subset of $V$). In other words, $C(S)=\{s_1,\ldots,s_r\}$. When there is no confusion we write $S$ instead of $C(S)$. The travel time of a route $S=(s_j)_{j=0}^r$ corresponds to the value $T(S)=\sum_{j=1}^r[t(s_{j-1},s_j)+t_0]+t(s_r,D)$.

A set of $m$ routes $\widehat{S}=\{S^1,\ldots,S^m\}$ is feasible if $C(S^1)\dot\cup\ldots\dot\cup C(S^m)=V$. We define the makespan$(\widehat{S})=\max_{1\le k\le m}T(S^k)$. In the Multiple Traveling Salesman Problem with Handling Times (mTSPHT) the goal is to find a feasible set of $m$ routes that minimizes the makespan. In this work we study problem mTSPHT when the graph $G$ is a spider with the depot $D$ located at its center.

A branch $B$ is a maximal path in $G$ starting in the depot $D$. (Obviously, $B$ can be seen as a particular route). We denote by $M \geq 1$ the number of branches, and we assume that they are enumerated as $B_1, \ldots, B_M$. We can encode every vertex $v \in V \setminus \{D\}$ with two positive integers: (1) the branch $b$ where $v$ belongs, with $1 \leq b \leq M$, and (2) the *label* $j = \text{label}(v)$ defined as the distance from $D$ to $v$. We write $v = v_j^b$ to refer to the unique vertex with label $j$ and branch $b$.

Let $S^k$ be a route and let $B_b$ be a branch of the spider $G$. Let $v_{k,b}$ be the farthest node of the branch $B_b$ visited in $S^k$. If $S^k$ does not include any node of branch $B_b$, then $v_{k,b} = D$. It is clear that the optimal solution for visiting the clients of $C(S^k)$ is to go branch by branch (the order of the branches does not matter for the makespan), visiting all the clients until the farthest one, come back to the depot, and then start with another branch. Therefore, we are going to assume that this is indeed the route $S^k$ taken by the $k$-th salesperson. In other words, $T(S^k) = 2\sum_{b=1}^{M} t(D, v_{k,b}) + |S^k|t_0$.

**Proposition 1** mTSPHT *for stars (and thus, also for spiders) is* ***strongly NP-hard****. In fact, with $m = 2$ it is still* ***NP-hard****.*

*Proof.* We show a reduction from the *Multiprocessor Scheduling Problem* (MSP) to mTSPHT. MSP is a **strongly NP-hard** problem when $m$ is part of the input, and remains **NP-hard** even for 2 available machines [6]. MSP is defined as follows. There are $N$ jobs $J_1, \ldots, J_N$, each job $J_i$ requires a processing time $t_i$. We have to schedule these $N$ jobs in $m$ machines in such a way that the makespan (the largest completion time) is minimized. The reduction is as follows.

- We construct a star $G$ with a depot $D$ at its center and $N$ leaves connected to $D$, each leaf corresponding to a job $J_i$.
- We define $t_0$, the handling time, as the shortest processing time. Therefore, $t_0 = t_z$ for some particular $1 \leq z \leq N$.
- We define the travel time associated to each edge $e = \{D, J_i\}$ as $t(e) = \frac{t_i - t_z}{2}$.
- There are $m$ salespersons (corresponding to the $m$ machines).

Observe that the output of any instance of MSP and the output of the corresponding instance of mTSPHT, created through the aforementioned reduction, are the same. It follows that the total processing time of machine $k$ is exactly $T(S^k)$. Therefore, minimizing the makespan in MSP is equivalent to minimizing the makespan in mTSPHT. □

Recall that we are always assuming that any route $S^k$ goes branch by branch, visiting all the clients until the farthest one in $C(S^k)$, comes back to the depot, and then start with another branch. We now explain how to transform any of these solutions into a more structured one, without increasing the makespan.

Let us define $S^{k,b}$ as the clients of the $k$-th route that are in branch $b$. Consider first the situation where a branch $B_b$ is visited by only two salespersons, through routes $S^i$ and $S^j$. Let $n_b$ be the size of the branch. Assume, without loss of generality, that the farthest node of the branch is visited by the $i$-th salesperson. More precisely: $v_{i,b} = v_{n_b}^b \in S^{i,b}$. Now, the $i$-th salesperson and

the $j$-th salesperson exchange clients (See Figure 1) in such a way that the number of clients served by each does not change and, all the clients served by the $j$-th salesperson are closer to the depot than the clients served by the $i$-th salesperson. Since the time spent by each salesperson in the branch depends only on the number of clients visited and the distance of the farthest one, the previous modification does not increase the makespan. If there are more than two routes in a branch we do exactly the same, shifting clients between pair of routes (see Figure 2) until the set of clients served by each salesperson on each branch is a consecutive block. We say that such type of solutions is *structured in intervals*. Therefore:

**Proposition 2** mTSPHT *in the spider admits an optimal solution that is structured in intervals.*

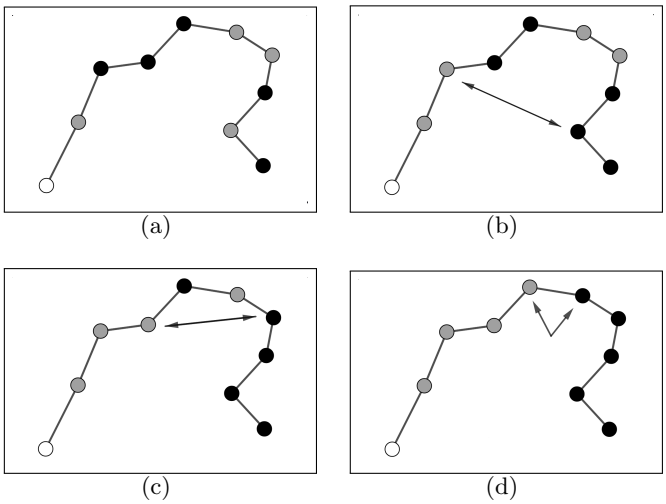
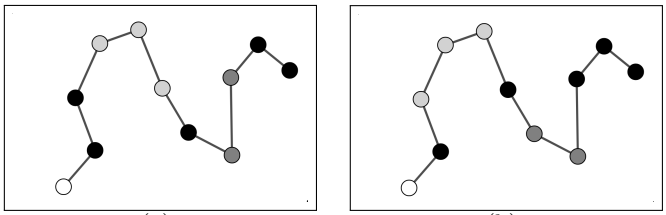



**Fig. 1.** Transforming an arbitrary solution of a branch into a solution structured in intervals. In this case there are two routes, one represented by grey nodes and the other by black nodes; the depot is white.

## 3 Canonical Solutions

We already know that there exists an optimal solution that is structured in intervals. In this section we will prove the existence of optimal solutions with even more structure. For this end we need some definitions.

**Definition 3** ***The $\preccurlyeq$ relation.*** *Let $B_b$ be a branch of a spider $G = (V, E)$. Let $S^i$ and $S^j$ be two routes. We say that $S^{i,b} \preccurlyeq S^{j,b}$ if the clients of $S^{i,b}$ are all*



(a) (b)

**Fig. 2.** In this example there are three salespersons visiting the clients of the branch. By shifting the assignment of two clients we get closer to a solution structured in intervals, without increasing the makespan.

*closer to the depot than those of* $S^{j,b}$*. Formally,* $\forall u \in C(S^{i,b}), \forall v \in C(S^{j,b})$ : $\text{label}(u) < \text{label}(v)$*. We also write* $S^i \preccurlyeq S^j$ *if* $S^{i,b} \preccurlyeq S^{j,b}$ *holds for every branch* $B_b$ *with* $1 \leq b \leq M$*.*

**Definition 4** ***Concurrent routes.*** *Let* $G = (V, E)$ *be a spider. Let* $S^i$ *and* $S^j$ *be two routes. We say that* $S^i$ *and* $S^j$ *are concurrent if there exists at least one branch that they both visit (see Figure 3).*

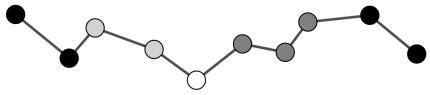


**Fig. 3.** There are three routes: black, dark gray and light gray. The depot is white. The black route is concurrent with the two others. But the light gray route and the dark gray route are not.

**Definition 5** ***Antisymmetric solutions.*** *Let* $G = (V, E)$ *be a spider. We say that a solution (a collection of* $m$ *routes) is antisymmetric if, for every two concurrent routes* $S^i$ *and* $S^j$*, either* $S^i \preccurlyeq S^j$ *or* $S^j \preccurlyeq S^i$*.*

**Proposition 6** *The* mTSPHT *problem for spiders admits an optimal solution that is antisymmetric.*

*Proof.* Let us suppose that there is no optimal antisymmetric solution. Let us define an obstruction as a 4-tuple $(B_x, B_y, S^i, S^j)$ such that $S^i$ and $S^j$ visit both $B_x$ and $B_y$, $S^{i,x} \preccurlyeq S^{j,x}$ and $S^{j,y} \preccurlyeq S^{i,y}$. Clearly, a solution is antisymmetric iff there are no obstructions. Consider an optimal solution with a minimum number of obstructions. Let $(B_x, B_y, S^i, S^j)$ be an obstruction for such solution (in Figure 4(a) these are the black and dark gray routes). In order to get a

contradiction we perform local changes in such a way that: (1) we do not create new obstructions, (2) we do not increase the makespan, (3) $(B_x, B_y, S^i, S^j)$ is not an obstruction anymore.

The local changes are defined as follows. First, consider the branch $B_x$. If $S^{i,x}$ and $S^{j,x}$ are consecutive intervals (this happens in the right branch of Figure 4(a)), then assign the last client (the one with largest label) visited by the route $S^{i,x}$ to the route $S^{j,x}$ (right branch of Figure 4(b)).

If $S^{i,x}$ and $S^{j,x}$ are not consecutive intervals, we shift the assignment of all nodes between $S^{i,x}$ and $S^{j,x}$ one step towards the depot. With this movement, $S^i$ looses the farthest client visited (by the $i$-th salesperson) in $B_x$. On the other hand, one client is left unassigned: the client with smallest label in $S^{j,x}$ (because such assignment was shifted towards the depot). We simply assign this client to $S^j$. Now, in the branch $B_x$, there is one more client visited by $S^j$ and one less client visited by $S^i$.

We do the same in $B_y$, changing the roles of $S^i$ and $S^j$ (see the development of the left branch in Figure 4). We iterate these local changes until the 4-tuple $(B_x, B_y, S^i, S^j)$ is not an obstruction anymore. This will end up happening because at every step both $|S^{i,x}|$ and $|S^{j,y}|$ decrease by one: at some point, one of them will be zero (see Figure 4(d)). □

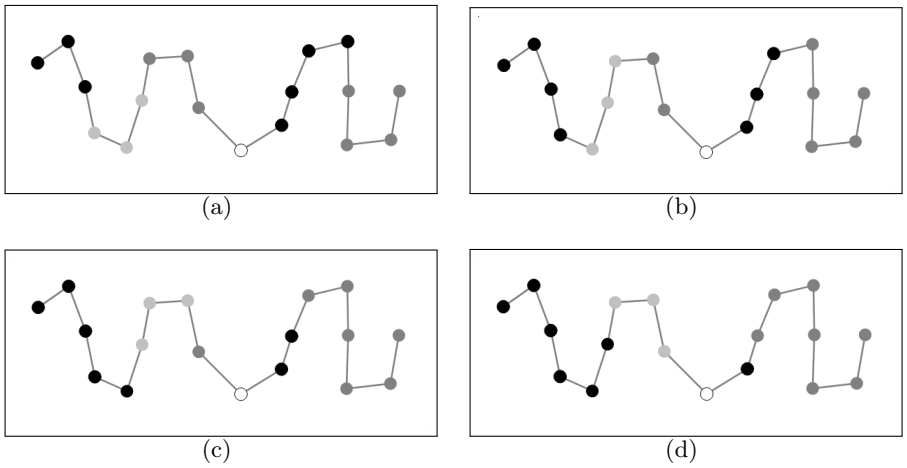



**Fig. 4.** Eliminating an obstruction.

**Definition 7** ***Transitive solutions.*** *Let $G = (V, E)$ be a spider. We say that a solution (a collection of $m$ routes) is transitive if, for every three routes $S^i, S^j, S^k$ such that $S^i \preccurlyeq S^j$ and $S^j \preccurlyeq S^k$, it follows that, if $S^i$ and $S^k$ are concurrent, then $S^i \preccurlyeq S^k$.*

**Definition 8** ***Well-structured solutions.*** *Let $G = (V, E)$ be a spider. We say that a solution (a collection of $m$ routes) is well-structured if it is antisymmetric and transitive.*

**Proposition 9** *The* mTSPHT *problem for spiders admits an optimal solution that is well-structured.*

*Proof.* The existence of an optimal antisymmetric solution has already been proven. Of these solutions, consider the one with the fewest number of 6-tuples $(B_x, B_y, B_z, S^i, S^j, S^k)$ such that $S^i \preccurlyeq S^j$, $S^j \preccurlyeq S^k$ and $S^k \preccurlyeq S^i$, $B_x$ is visited by (at least) $S^i$ and $S^j$, $B_y$ is visited by $S^j$ and $S^k$ and $B_z$ is visited by $S^i$ and $S^k$. We call these 6-tuples obstructions. Consider the following local change: if $S^i$ and $S^j$ are not consecutive in $B_x$, in the same way as we did in the proof of Proposition 6, we shift each route between $S^{i,x}$ and $S^{j,x}$ towards the depot. If there are no routes between $S^{i,x}$ and $S^{j,x}$ (they are consecutive) we just unassign the client of largest label in $S^{j,x}$. In both cases, only the last client of $S^{j,x}$ is left unassigned. We do the same in the other two branches (cyclically exchanging roles). Note that $S^{i,x}$, $S^{j,y}$ and $S^{k,z}$ all lose a client. We then assign the unassigned client of $B_x$ to $S^j$, the unassigned client of $B_y$ to $S^k$, and the unassigned client of $B_z$ to $S^i$. The number of clients that each salesperson visits is the same and the distance from the depot to the last client of each route (in each branch) does not increase. Then, this new solution is still optimal. The idea is to repeat this local change in the 6-tuple $(B_x, B_y, B_z, S^i, S^j, S^k)$ as long as possible. The process ends since all $|S^{i,x}|, |S^{j,y}|$ and $|S^{k,z}|$ decrease by one and, at the end, an optimal antisymmetric solution is obtained with one less obstruction, a contradiction. Hence, the minimum number of obstructions is zero. □

**A polynomial Dynamic Programming algorithm for a fixed number of branches.** A well-structured solution has the following property: there is always one salesperson who visits just blocks of consecutive clients located at the end of some branches; once we remove such salesperson together with the clients he/she visited, the remaining (smaller) solution is also well-structured. Every instance $\mathcal{I}$ can be characterized by the travel time $t$, the handling time $t_0$, the number of salespersons $m$ and the (ordered) set of $M$ branches $\widehat{B}$. Let $F_{\mathcal{I}}(k_1, \ldots, k_M, m')$ be the optimum makespan for the subinstance in which $m'$ salespersons must collectively visit only the first $k_i$ clients of branch $B_i$. The solution of mTSPHT, $F_{\mathcal{I}}(|B_1|, \ldots, |B_M|, m)$, can be computed by dynammic programming (DP) as follows:

- For all $m' \geq 1$ $F_{\mathcal{I}}(0, \ldots, 0, m') = 0$.
- if the $k_i$ are not all zero, $F_{\mathcal{I}}(k_1, \ldots, k_M, 1) = \sum_{b=1}^{M} (2t(D, v_{k_b}^b) + t_0 k_b)$.
- if the $k_i$ are not all zero and $m' > 1$, $F_{\mathcal{I}}(k_1, \ldots, k_M, m')$ is the minimum, for all possible values of $0 \leq i_1 \leq k_1, \ldots, 0 \leq i_M \leq k_M$ of

$$\max \left\{ F_{\mathcal{I}}(k_1 - i_1, \ldots, k_M - i_M, m' - 1),\ \sum_{b=1}^{M} (2t(D, v_{k_b}^b) + t_0 i_b) \mathbb{1}_{\{i_b > 0\}} \right\}.$$

Since the DP table has size at most $mN^M$, and each value is the minimum among at most $N^M$ values, the time complexity of this DP is at most $O(mN^{2M})$.

## 4 PTAS for a fixed number $m$ of salespersons

We start by defining the *weight* of a branch $B_b$ of the spider as the time it would take a single salesman to process it alone: $\text{weight}(B_b) = 2d(D, v^b_{|B_b|}) + t_0|B_b|$. Also, for a set of branches $\widehat{B}$, define $\text{weight}(\widehat{B}) = \sum_{B\in\widehat{B}} \text{weight}(B_b)$. Define $H_m = \frac{1}{m}\sum_{b=1}^{M} \text{weight}(B_b) = \frac{1}{m}(Nt_0 + 2\sum_{B\in\widehat{B}} d(D, v^b_{|B_b|}))$. Any feasible solution $\widehat{S} = \{S^1, \ldots, S^m\}$ has to cover the distance from the depot to the client of largest label in each branch twice, and each client has to be visited. Therefore, $\sum_{k=1}^{m} T(S^k) \geq mH_m$ and $\text{makespan}(\widehat{S}) = \max_{k\in\{1,\ldots,m\}} T(S^k) \geq H_m$. In particular, $H_m$ is a lower bound for the makespan (i.e. for mTSPHT),

Our goal is to solve mTSPHT through binary search, using the DP defined at the end of previous section.

**Theorem 10** *There exists a PTAS for problem* mTSPHT *that finds a* $(1+\varepsilon)$*-approximation and takes time* $N^{O(m/\varepsilon)}$.

In what remains we prove previous theorem. We divide this proof in three parts: the algorithm, the complexity analysis, and the proof of correctness.

### 4.1 The Algorithm

Consider an instance $\mathcal{I}$ and a tolerance $\varepsilon > 0$. Sort the branches according to their weight in decreasing order, define a threshold $K = \lceil m/\varepsilon \rceil$, and build a new instance $\mathcal{I}'$ as follows. The first $K$ branches are not modified. The remaining branches are grouped so that the sum of the weights on each group does not exceed $mH_m/K$. Each of these groups is identified with a new branch consisting of a single client. The new instance has at most $3K$ branches, and it is solved using the DP of last section. We transform the output into a solution of the original instance, with a small "price" to pay. Such price explains the $(1+\varepsilon)$ term of the approximation algorithm. Formally:

1. *Initialization*
   - Compute the weight of each branch in $\mathcal{I}$ and compute $H_m$.
   - Sort the branches in decreasing order according to their weights.
   - Define the threshold $K = \lceil m/\varepsilon \rceil$.
2. *The Simple Case*
   - If $M \leq K$, solve $F_{\mathcal{I}}(|B_1|, \ldots, |B_M|, m)$ using the DP.
   - If $M > K$ continue with next step.
3. *Grouping the Branches*
   - From the $(K+1)$-th branch on, i.e., from $B_{K+1}$, group the branches in a greedy way so that the weight of each group is at most $mH_m/K$.
   - Replace each group of branches $C$ by a branch with a single client $v_C$, and set $t(D, v_C) = (\text{weight}(C) - t_0)/2$.
   - Denote by $\mathcal{I}'$ the new instance with branches $B'_1, \ldots, B'_{M'}$ (the $K$ first ones of $\mathcal{I}$, together with the new, singleton branches).

4. *Solving the Problem for the Modified Instance*
   - Solve $F_{\mathcal{I}'}(|B'_1|, \ldots, |B'_{M'}|, m)$ through the DP.
   - Call $\widehat{R} = \{R^1, \ldots R^m\}$ the set of routes corresponding to the solution.
5. *Solving the Problem for the Original Instance*
   For each branch $B'_b$ of instance $\mathcal{I}'$ and each salesperson $k$:
   - If $b \leq K$, then $S^{k,b} = R^{k,b}$.
   - If $b > K$, then the branch is a singleton $v_C$. If, according to the solution $\widehat{R}$, the $k$-th salesperson serves $v_C$, then we assign to $S^k$ all the clients of all the branches grouped in $v_C$.

### 4.2 Complexity

Computing weights and sorting the branches takes time $O(N) + O(M \log M) = O(N \log N)$. If $M \leq K$, then solving the DP takes time $O(mN^{2K})$. Since $m \leq N$, this can be written as $O(N^{2K+1})$.

If $M > K$, then we need first to group the branches. There can be at most $2K$ new groups. Therefore, since $M \leq N$, generating the modified instance takes time $O(N^2)$. Note that the new number of branches $M' \leq 3K$. Hence, solving the new instance's DP takes time $O(mN^{2(3K)}) = O(mN^{6K}) = O(N^{6K+1})$.

Finally, since there are at most $3K$ branches in the modified instance, obtaining the approximate solution in step 5 takes time $O(mN^{3K})$.

Overall, the time complexity of the whole algorithm is dominated by the term $O(N^{6K+1}) = O(N^{6(m/\varepsilon)+7}) = N^{O(m/\varepsilon)}$.

### 4.3 Correctness

If the number of branches $M$ is less or equal than $K$ we are in *The Simple Case* and the problem is solved to optimality. We may therefore assume that $M > K$, and the algorithm creates a new instance $\mathcal{I}'$ by *Grouping the Branches*. Since the branches of the original instance are sorted by weight and since the total weight is $mH_m$, the $K$-th branch (and each of the following branches) weights at most $mH_m/K$. This implies that at most $2K$ groups are formed. In fact, if there were $2K + 1$ groups (or more), the weight of at least two groups $C_1$ and $C_2$ should be smaller than $mH_m/2K$. This is not possible, because if we created $C_1$ before $C_2$, then the weight of any branch sorted after those in $C_1$ is smaller than $mH_m/2K$, and at least one could be added to $C_1$ without exceeding the threshold. Therefore, the number $M'$ of branches of the modified instance is at most $K + 2K = 3K$. The travel time from the depot $D$ to any "new" client $v_C$ is chosen so that the weight of the new branch associated with $v_C$ is equal to the original weight of the branches in $C$.

After *Grouping the branches*, we *Solve the Problem for the Modified Instance*, and we obtain the optimal solution $\widehat{R}$ of this modified instance. Finally, we *Solve the Problem for the Original Instance* in order to obtain a solution $\widehat{S}$ of the original instance.

*Claim.* If $\widehat{S}_{\mathrm{OPT}} = (S^1_{\mathrm{OPT}}, \ldots, S^m_{\mathrm{OPT}})$ is an optimal solution of the original instance, then there exists a solution $\widehat{R}_{\mathrm{mod}} = (R^1_{\mathrm{mod}}, \ldots, R^m_{\mathrm{mod}})$ of the modified instance such that, for each saleperson $k$, $T(R^k_{\mathrm{mod}}) \leq T(S^k_{\mathrm{OPT}}) + mH_m/K$.

*Proof.* Let $\widehat{B} = \{B_{K+1}, B_{K+2}, \ldots, B_M\}$ be the branches of the original instance that were grouped into new branches $\widehat{B}' = \{B'_{K+1}, \ldots, B'_{M'}\}$ of the modified instance. Let $T(S^{k,b}_{OPT}) = 2t(D, v_{k,b}) + |B_b|t_0$ be the time spent by the $k$-th salesperson of the optimal at $B_b$. Let $A_k = \sum_{b=K+1}^{M} T(S^{k,b}_{OPT})$.

Consider any partition $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ of the branches of $\widehat{B}'$ into $m$ parts (allowing empty parts). For $k \in \{1, \ldots, m\}$ we define the utility of $k$ as $U_{\mathcal{Q}}(k) = \left(\sum_{B' \in Q_k} \mathrm{weight}(B')\right) - A_k$. We can use $\mathcal{Q}$ to create a solution $\widehat{R}_{\mathrm{mod}}$ of the modified instance. In this solution, the $k$-th salesperson serves the clients of $\{B_1, \ldots, B_K\}$ according to $\widehat{S}_{\mathrm{OPT}}$ and, in addition, he/she also serves all branches of $Q_k$. With this, the total time used by the $k$-th salesperson is $T(R^k_{\mathrm{mod}}) = T(S^k_{OPT}) - A_k + \sum_{B' \in Q_k} \mathrm{weight}(B') = T(S^k_{\mathrm{OPT}}) + U_{\mathcal{Q}}(k)$. To conclude we prove the existence of a partition $\mathcal{Q}$ such that $U_{\mathcal{Q}}(k) \leq mH_m/K$ for all $k$.

Of all the possible partitions $\mathcal{Q}$ consider the one that (1) minimizes the maximum utility $\eta_{\mathcal{Q}} = \max_{k \in \{1,\ldots,m\}} U_{\mathcal{Q}}(k)$ and, subject to this, (2) minimizes the number of indexes $k$ with maximum utility (i.e, such that $U_{\mathcal{Q}}(k) = \eta_{\mathcal{Q}}$).

Define $A := \sum_{B \in \widehat{B}} \mathrm{weight}(B)$, and note that $A \leq \sum_{k=1}^{m} A_k$, since in the sum on the right, each edge of the branches of $\widehat{B}$ can be counted more than twice, while, in $A$, each edge is counted only 2 times. On the other hand, $A$ can also be computed as $A = \sum_{k=1}^{m} \sum_{B' \in Q_k} \mathrm{weight}(B')$.

It follows that $\sum_{k=1}^{m} U_{\mathcal{Q}}(k) = \sum_{k=1}^{m} \left(\sum_{B' \in Q_k} \mathrm{weight}(B')\right) - A_k \leq 0$. Therefore, there is an index $j$ such that $U_{\mathcal{Q}}(j) \leq 0$. Let us then consider an index $i$ such that $\eta_{\mathcal{Q}} = U_{\mathcal{Q}}(i)$ and suppose, by contradiction, that $U_{\mathcal{Q}}(i) > mH_m/K$.

We could create a new partition $\mathcal{Q}'$, starting from $\mathcal{Q}$, reassigning the branch $B' \in Q_i$ of greater weight to the set $Q_j$. With this, the utility of $i$ decreases by $\mathrm{weight}(B') > 0$, and the utility of $j$ increases by $\mathrm{weight}(B')$. Recalling that $\mathrm{weight}(B') < mH_m/K$ (in this way the branches of the modified instance were built), we conclude that either the maximum utility $\eta_{\mathcal{Q}'}$ of the new partition is less than $\eta_{\mathcal{Q}'}$, or the maximum utility remains, but in $\mathcal{Q}'$, the number of indexes of maximum utility decreases (since $i$ does not have maximum utility anymore). In any case we come to a contradiction with the choice of $\mathcal{Q}$. □

Since $\widehat{R}$ is the optimal solution of the modified instance it follows that $\mathrm{makespan}(\widehat{R}) \leq \mathrm{makespan}(\widehat{R}_{\mathrm{mod}})$. Therefore, using previous claim, we have that $\mathrm{makespan}(\widehat{R}) \leq \mathrm{makespan}(\widehat{S}_{\mathrm{OPT}}) + mH_m/K$.

But, as explained in the beginning of this section, $H_m$ is a lower bound for the makespan of every feasible solution. Hence,

$$\mathrm{makespan}(\widehat{R}) \leq \mathrm{makespan}(\widehat{S}_{\mathrm{OPT}}) \left(1 + \frac{m}{K}\right).$$

Note that, by construction, the makespan of the solution $\widehat{S}$ given by the algorithm is exactly the makespan of $\widehat{R}$. Therefore,

$$\text{makespan}(\widehat{S}) = \text{makespan}(\widehat{R}) \leq \text{makespan}(\widehat{S}_{\text{OPT}}) \left(1 + \frac{m}{K}\right).$$

By recalling that $K = \lceil \frac{m}{\varepsilon} \rceil$, we conclude that $\widehat{S}$ is indeed a $(1+\varepsilon)$-appoximation.

## References


1. Xiaoguang Bao and Zhaohui Liu. Approximation algorithms for single vehicle scheduling problems with release and service times on a tree or cycle. *Theor. Comput. Sci.*, 434:1–10, 2012.
2. Amariah Becker and Alice Paul. A framework for vehicle routing approximation schemes in trees. In *Workshop on Algorithms and Data Structures*, pages 112–125. Springer, 2019.
3. Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.
4. Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.
5. Greg N Frederickson, Matthew S Hecht, and Chul E Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.
6. Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
7. Daya Gaur, Arvind Gupta, and Ramesh Krishnamurti. A 5/3-approximation algorithm for scheduling vehicles on a path with release and handling times. *Inf. Process. Lett.*, 86:87–91, 04 2003.
8. Martin Grötschel, Michael Jünger, and Gerhard Reinelt. Optimal control of plotting and drilling machines: a case study. *Zeitschrift für Operations Research*, 35(1):61–84, 1991.
9. Yoshiyuki Karuno and Hiroshi Nagamochi. A 2-approximation algorithm for the multi-vehicle scheduling problem on a path with release and handling times. In Friedhelm Meyer auf der Heide, editor, *Algorithms — ESA 2001*, pages 218–229, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
10. Yoshiyuki Karuno and Hiroshi Nagamochi. An approximability result of the multi-vehicle scheduling problem on a path with release and handling times. *Theoretical Computer Science*, 312:267–280, 01 2004.
11. Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358, 1992.
12. Eugene L Lawler. The traveling salesman problem: a guided tour of combinatorial optimization. *Wiley-Interscience Series in Discrete Mathematics*, 1985.
13. H Donald Ratliff and Arnon S Rosenthal. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983.
14. Liang Xu, Zhou Xu, and Dongsheng Xu. Exact and approximation algorithms for the minmax k-traveling salesmen problem on a tree. *European Journal of Operational Research*, 227(2):284–292, 2013.

15. Zhou Xu and Liang Xu. Approximation algorithms for min-max path cover problems with service handling time. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation*, pages 383–392, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
16. Wei Yu and Zhaohui Liu. Vehicle routing problems on a line-shaped network with release time constraints. *Oper. Res. Lett.*, 37:85–88, 03 2009.