



A PTAS for Capacitated Vehicle Routing on Trees

Claire Mathieu  

CNRS Paris, France

Hang Zhou  

École Polytechnique, Institut Polytechnique de Paris, France

Abstract

We give a polynomial time approximation scheme (PTAS) for the unit demand capacitated vehicle routing problem (CVRP) on trees, for the entire range of the tour capacity. The result extends to the splittable CVRP.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases approximation algorithms, capacitated vehicle routing, graph algorithms, combinatorial optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.95

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: <https://arxiv.org/abs/2111.03735>

Funding This work was partially funded by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR).

1 Introduction

Given an edge-weighted graph with a vertex called *depot*, a subset of vertices with demands, called *terminals*, and an integer *tour capacity* k , the *capacitated vehicle routing problem* (CVRP) asks for a minimum length collection of tours starting and ending at the depot such that those tours together cover all the demand and the total demand covered by each tour is at most k . In the *unit demand* version, each terminal has unit demand, which is covered by a single tour;¹ in the *splittable* version, each terminal has a positive integer demand and the demand at a terminal may be covered by multiple tours.

The CVRP was introduced by Dantzig and Ramser in 1959 [15] and is arguably one of the most important problems in Operations Research. Books have been dedicated to vehicle routing problems, e.g., [26, 18, 14, 3]. Yet, these problems remain challenging, both from a practical and a theoretical perspective.

Here we focus on the special case when the underlying metric is a tree. That case has been the object of much research. The splittable tree CVRP was proved NP-hard in 1991 [24], so researchers turned to approximation algorithms. Hamaguchi and Katoh [21] gave a simple lower bound: every edge must be traversed by enough tours to cover all terminals whose shortest paths to the depot contain that edge. Based on this lower bound, they designed a 1.5-approximation in polynomial time [21]. The approximation ratio was improved to $(\sqrt{41} - 1)/4$ by Asano, Katoh, and Kawashima [4] and further to $4/3$ by Becker [6], both results again based on the lower bound from [21]. On the other hand, it was shown in [4] that using this lower bound one cannot achieve an approximation ratio better than $4/3$. More recently, researchers tried to go beyond a constant factor so as to get a $(1 + \epsilon)$ -approximation, at the cost of relaxing some of the constraints. When the tour capacity is allowed to be

¹ Thus we may identify the demand covered with the number of terminals covered.



violated by an ϵ fraction, there is a bicriteria PTAS for the unit demand tree CVRP due to Becker and Paul [10]. When the running time is allowed to be quasi-polynomial, Jayaprakash and Salavatipour [22] very recently gave a *quasi-polynomial time approximation scheme (QPTAS)* for the unit demand and the splittable versions of the tree CVRP. In this paper, we close this line of research by obtaining a $(1 + \epsilon)$ -approximation without relaxing any of the constraints – in other words, a *polynomial-time approximation scheme (PTAS)*.

► **Theorem 1.** *There is an approximation scheme for the unit demand capacitated vehicle routing problem (CVRP) on trees with polynomial running time.*

► **Corollary 2.** *There is an approximation scheme for the splittable capacitated vehicle routing problem (CVRP) on trees with running time polynomial in the number of vertices n and the tour capacity k .*

To the best of our knowledge, this is the first PTAS for the CVRP in a non-trivial metric and for the entire range of the tour capacity. Previously, PTASs for small capacity as well as QPTASs were given for the CVRP in several metrics, see Section 1.1.3.

1.1 Related Work

Our algorithms build on [22] and [10] but with the addition of significant new ideas, as we now explain.

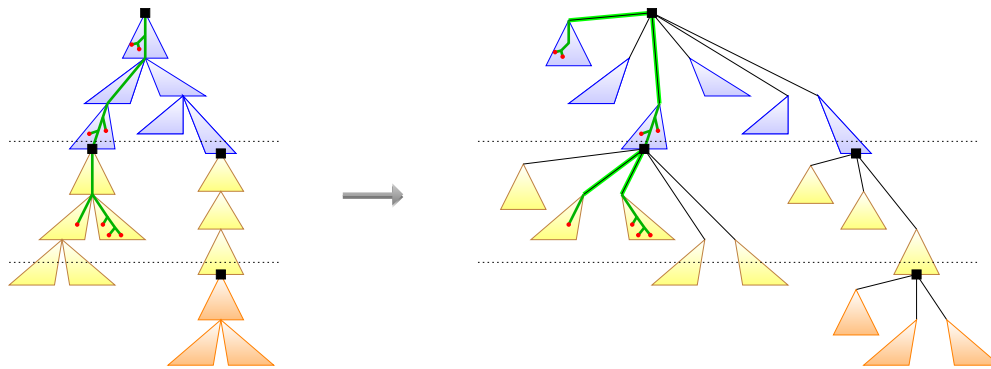
1.1.1 Comparison with the QPTAS in [22]

Jayaprakash and Salavatipour noted in [22] that

“*it is not clear if it (the QPTAS) can be turned into a PTAS without significant new ideas.*”

The running time in [22] is $n^{O_\epsilon(\log^4 n)}$. Where do those four $\log n$ factors in the exponent come from? At a high level, the QPTAS in [22] consists of three parts: (1) reducing the height of the tree; (2) designing a bicriteria QPTAS; (3) going from the bicriteria QPTAS to a true QPTAS. Our approach builds on [22] but differs from it in each of the three parts, so that in the end we get rid of all of four $\log n$ factors, hence a PTAS.

- (1) Jayaprakash and Salavatipour [22] reduce the input tree height from $O(n)$ to $O_\epsilon(\log^2 n)$; whereas instead of the input tree, we consider a *tree of components* (Lemma 9) and reduce its height to $O_\epsilon(1)$, see Figure 1. Pleasingly, the height reduction (Section 4) is much simpler than in [22]. The analysis differs from [22] and uses the structure of a near-optimal solution established in Section 3 and the *bounded distance* property (Definition 3 and Theorem 5).
- (2) In the *adaptive rounding* used in [22], they consider the entire range $[1, k]$ of the demands of subtours and partition the subtour demands into *buckets*, resulting in $\Omega_\epsilon(\log k)$ different subtour demands after rounding. In our approach, we define *large* and *small* subtours inside components, depending on whether their demands are $\Omega_\epsilon(k)$ (Definition 14). Then we transform the solution structure to eliminate small subtours (Section 3), hence only $O_\epsilon(1)$ different subtour demands after rounding. This elimination requires a delicate handling of small subtours. Thanks to the additional structure, our analysis of the adaptive rounding is simpler than in [22], and in particular, we do not need the concept of buckets.



■ **Figure 1** Height reduction for a tree of components. The left figure represents the initial tree of components, where each triangle represents a component. We partition the components into classes (indicated by blue, yellow, and orange), according to the distances from the roots of the components to the root of the tree, and we reduce the height within each class to 1 (right figure), see Section 4. The thick green path in the left figure represents a tour in an optimal solution. The red circular nodes are the terminals visited by that tour. The corresponding tour in the new tree (right figure) spans the same set of terminals.

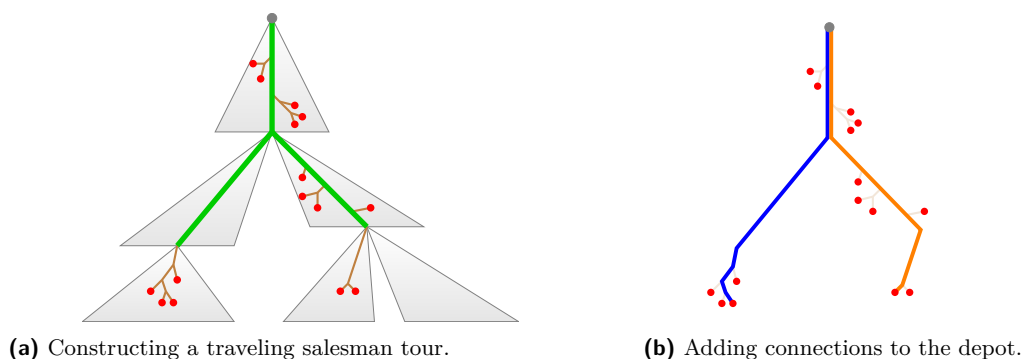
- (3) Jayaprakash and Salavatipour show that the *orphan tokens*, which are removed from the tours exceeding capacity, can probably be covered by duplicating a small random set of tours in the optimal solution. Their approach requires remembering the demands of $\Omega(\log n)$ subtours passing through each edge. To avoid this $\Omega(\log n)$ factor, our approach to cover the orphan tokens (Section 3.1) is different, see Figure 2. The analysis of our approach (Sections 3.2 and 3.3) contains several novelties of this paper.

1.1.2 Comparison with the Bicriteria PTAS in [10]

Why is the algorithm in [10] a bicriteria PTAS, but not a PTAS?

Becker and Paul [10] start by decomposing the tree into *clusters*. (1) They require that each *leaf* cluster is visited by a single tour. When the violation of the tour capacity is not allowed, this requirement does not preserve a $(1 + \epsilon)$ -approximate solution. (2) They also require that each *small* internal cluster is visited by a single tour. To that end, they modify the optimal solution by reassigning all terminals of a small cluster to some existing tour at the cost of possibly violating the tour capacity. Such modifications do not seem achievable in the design of a PTAS.

In this paper, we start by defining *components* (Lemma 9), inspired by *clusters* in [10]. Unlike [10], we allow terminals in any component to be visited by multiple tours. However, allowing many subtours inside a component could result in an exponential running time for a dynamic program. To prevent that, we modify the solution structure inside a component so that the number of subtours becomes bounded (Theorem 13). Instead of considering all subtours simultaneously as in [10], we distinguish *small* subtours from *large* subtours (Definition 14). Inspired by [10], we combine small subtours and reallocate them to existing tours such that the violation of the tour capacity is an $O(\epsilon)$ fraction, see Steps 1 to 3 of the construction in Section 3.1. Next, we use the *iterated tour partitioning (ITP)* and its postprocessing to reduce the demand of the tours exceeding capacity (Figure 2), which is a novelty in this paper, see Steps 4 to 6 of the construction in Section 3.1. The ITP algorithm and its postprocessing are analyzed in Sections 3.2 and 3.3. In particular, we bound the cost due to the ITP algorithm thanks to the *bounded distance* property (Theorem 5) and to



■ **Figure 2** Covering the *orphan tokens* (in red). In Figure 2a, the orphan tokens are contained in the small pieces (in brown) that are removed from the tours exceeding capacity. We add the thick paths (in green) to connect all of the small pieces to the root of the tree. The cost of the thick paths is an $O(\epsilon)$ fraction of the optimal cost (Lemma 17 and Corollary 18), thanks to the bounded distance property. The induced traveling salesman tour is a double cover of the tree spanning the orphan tokens. Next, we apply the *iterated tour partitioning* (ITP) algorithm on that tour. In Figure 2b, the two paths (in blue and in orange) represent the connections to the depot added by the ITP algorithm. Their cost is again an $O(\epsilon)$ fraction of the optimal cost (Lemma 19), thanks to the bound on the number of orphan tokens and the bounded distance property.

the parameters in our component decomposition that are different from those in [10], see Remark 10. Besides the above novelties in our approach, the height reduction (Figure 1, see also Section 4), the adaptive rounding (Section 5), the reduction to bounded distances, as well as part of the dynamic program are new compared with [10]. These additional techniques are essential in the design of our PTAS, because of the more complicated solution structure inside components in our approach compared with the solution structure inside clusters in [10].

1.1.3 Other Related Work

Constant-factor approximations in general metric spaces

The CVRP is a generalization of the *traveling salesman problem* (TSP). In general metric spaces, Haimovich and Rinnooy Kan [20] introduced a simple heuristics, called *iterated tour partitioning* (ITP). Altinkemer and Gavish [2] showed that the approximation ratio of the ITP algorithm for the unit demand and the splittable CVRP is at most $1 + (1 - \frac{1}{k}) C_{\text{TSP}}$, where $C_{\text{TSP}} \geq 1$ is the approximation ratio of a TSP algorithm. Bompadre, Dror, and Orlin [12] improved this bound to $1 + (1 - \frac{1}{k}) C_{\text{TSP}} - \Omega(\frac{1}{k^3})$. The ratio for the unit demand and the splittable CVRP on general metric spaces was recently improved by Blauth, Traub, and Vygen [11] to $1 + C_{\text{TSP}} - \epsilon$, for some small constant $\epsilon > 0$.

QPTASs

Das and Mathieu [16] designed a QPTAS for the CVRP in the Euclidean space; Jayaprakash and Salavatipour [22] designed a QPTAS for the CVRP in trees and extended that algorithm to QPTASs in graphs of bounded treewidth, bounded doubling or highway dimension. When the tour capacity is fixed, Becker, Klein, and Saulpic [7] gave a QPTAS for planar graphs and bounded-genus graphs.

PTASs for small capacity

In the Euclidean space, there have been PTAS algorithms for the CVRP with small capacity k : work by Haimovich and Rinnooy Kan [20], when k is constant; by Asano et al. [5] extending techniques in [20], for $k = O(\log n / \log \log n)$; and by Adamaszek, Czumaj, and Lingas [1], when $k \leq 2^{\log^{f(\epsilon)}(n)}$. For higher dimensional Euclidean metrics, Khachay and Dubinin [23] gave a PTAS for fixed dimension ℓ and $k = O(\log^{\frac{1}{\ell}}(n))$. Again when the capacity is bounded, Becker, Klein and Schild [9] gave a PTAS for planar graphs; Becker, Klein, and Saulpic [8] gave a PTAS for graphs of bounded highway dimension; and Cohen-Addad et al. [13] gave PTASs for bounded genus graphs and bounded treewidth graphs.

Unsplittable CVRP

In the *unsplittable* version of the CVRP, every terminal has a positive integer demand, and the entire demand at a terminal should be served by a single tour. On general metric spaces, the best-to-date approximation ratio for the unsplittable CVRP is roughly 3.194 due to the recent work of Friggstad et al. [17]. For tree metrics, the unsplittable CVRP is APX-hard: indeed, it is NP-hard to approximate the unsplittable tree CVRP to better than a 1.5 factor [19] using a reduction from the bin packing problem. Labbé, Laporte and Mercure [24] gave a 2-approximation for the unsplittable tree CVRP. The approximation ratio for the unsplittable tree CVRP was improved to $(1.5 + \epsilon)$ very recently by Mathieu and Zhou [25], building upon several techniques in the current paper.

1.2 Overview of Our Techniques

The main part of our work focuses on the unit demand tree CVRP, and we extend our results to the splittable tree CVRP in the end of this work.

► **Definition 3** (bounded distances). *Let D_{\min} (resp. D_{\max}) denote the minimum (resp. maximum) distance between the depot and any terminal in the tree. We say that an instance has bounded distances if $D_{\max} < (\frac{1}{\epsilon})^{\frac{1}{\epsilon}-1} \cdot D_{\min}$.*

Theorem 1 follows directly from Theorems 4 and 5.

► **Theorem 4.** *There is a polynomial time $(1 + 4\epsilon)$ -approximation algorithm for the unit demand CVRP on the tree T with bounded distances.*

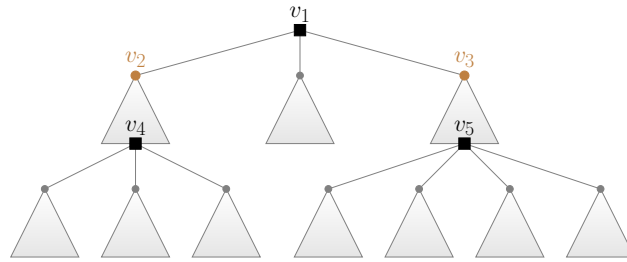
► **Theorem 5.** *For any $\rho \geq 1$, if there is a polynomial time ρ -approximation algorithm for the unit demand (resp. splittable, or unsplittable) CVRP on trees with bounded distances, then there is a polynomial time $(1 + 5\epsilon)\rho$ -approximation algorithm for the unit demand (resp. splittable, or unsplittable) CVRP on trees with general distances.*

Theorem 5 may be of independent interest for the splittable and the unsplittable versions of the tree CVRP.

Outline for unit demand instances with bounded distances (Theorem 4)

First, we show that there exists a near-optimal solution with a simple structure, and afterwards, we use a dynamic program to compute the best solution with that structure.

In Section 3, we consider the *components* of the tree T (Lemma 9) and we show that there exists a near-optimal solution such that terminals within each component are visited by a constant $O_{\epsilon}(1)$ number of tours and that each of those tours visits $\Omega_{\epsilon}(k)$ terminals in that component (Theorem 13). The proof of Theorem 13 contains several novelties in our



■ **Figure 3** At each vertex of the tree, the dynamic program memorizes the capacities used by the subtours in the subtree and their total cost. Terminals within each component are visited by $O_\epsilon(1)$ tours and each of those tours visits $\Omega_\epsilon(k)$ terminals in that component. Here is an example of the flow of execution in the dynamic program. First, independent computation in each component. Next, computation in the subtrees rooted at vertices v_4 and v_5 . Then computation for the subtrees rooted at vertices v_2 and v_3 . Finally, computation for the subtree rooted at vertex v_1 . The output is the best solution in the subtree rooted at v_1 . Vertices v_1, v_4 , and v_5 , whose degrees may be arbitrarily large, are where adaptive rounding of the subtour demands is needed to maintain polynomial time. The cost due to the rounding is small thanks to the way components are defined and the bounded distance property, and does not accumulate excessively because the height is bounded (Section 4).

work. We start by defining *large* and *small* subtours inside a component, depending on the number of terminals visited by the subtours. To construct a near-optimal solution with that structure, first, we detach small subtours from their initial tours, combine small subtours in the same component, and reallocate the combined subtours to existing tours. Then we remove subtours from tours exceeding capacity. To connect the removed subtours to the root of the tree, we include the *spines subtours* (Definition 12) of all *internal components*, and we obtain a traveling salesman tour. Next, we apply the *iterated tour partitioning (ITP)* algorithm on that tour, see Figure 2. Finally, in a postprocessing step, we eliminate the small subtours created due to the ITP algorithm. The complete construction is in Section 3.1; the feasibility of the construction is in Section 3.2; and the analysis on the constructed solution is in Section 3.3, which in particular uses the bounded distance property.

In Section 4, we transform the tree T into a tree \hat{T} that has $O_\epsilon(1)$ levels of components (Figure 1) and satisfies the following property.

► **Fact 6.** *The tree \hat{T} defined in Section 4 can be computed in polynomial time. The components in the tree \hat{T} are the same as those in the tree T . Any solution for the unit demand CVRP on the tree \hat{T} can be transformed in polynomial time into a solution for the unit demand CVRP on the tree T without increasing the cost.*

Thanks to the structure of the near-optimal solution on T (Section 3) and to the bounded distance property, the optimal cost for \hat{T} is increased by an $O(\epsilon)$ fraction compared with the optimal cost for T (Theorem 23).

In Section 5, we apply the *adaptive rounding* on the demands of the subtours in a near-optimal solution on \hat{T} . Recall that in the design of the QPTAS by Jayaprakash and Salavatipour [22], the main technique is to show the existence of a near-optimal solution in which the demand of a subtour can be rounded to the nearest value from a set of only *poly-logarithmic* threshold values. In our work, we reduce the number of threshold values to a *constant* $O_\epsilon(1)$ (Theorem 26). To analyze the adaptive rounding, observe that an extra cost occurs whenever we detach a subtour and complete it into a separate tour by connecting it to the depot. We bound the extra cost thanks to the structure of a near-optimal solution inside components (Section 3), the reduced height of the components (Section 4), as well as the bounded distance property.

In the full version of the paper, we design a *polynomial-time dynamic program* that computes the best solution on the tree \hat{T} that satisfies the constraints on the solution structure imposed by previous sections. The algorithm combines a dynamic program inside components and two dynamic programs in subtrees, see Figure 3. Thus we obtain the following Theorem 7.

► **Theorem 7.** *Consider the unit demand CVRP on the tree \hat{T} . There is a dynamic program that computes in polynomial time a solution with cost at most $(1 + 4\epsilon) \cdot \text{opt}$, where opt denotes the optimal cost for the unit demand CVRP on the tree T .*

Theorem 4 follows directly from Theorem 7 and Fact 6.

Reduction from general distances to bounded distances (Theorem 5)

In the full version of the paper, we prove Theorem 5. We use Baker’s technique to split tours into pieces such that each piece covers terminals that are within a certain range of distances from the depot. This requires duplicating some parts of the tours so that each piece of the tour is connected to the depot.

Extension to the splittable setting (Corollary 2)

In the full version of the paper, we extend the result in Theorem 1 to the splittable setting, thus obtaining Corollary 2.

Open questions

Previously, Jayaprakash and Salavatipour [22] extended their QPTAS on trees to QPTASs on graphs of bounded treewidth and beyond, including Euclidean spaces. While some of our techniques extend to those settings, others do not seem to carry over without significant additional ideas, so it is an interesting open question whether the techniques in our paper could be used in the design of PTAS algorithms for other metrics, such as graphs of bounded treewidth, planar graphs, and Euclidean spaces.

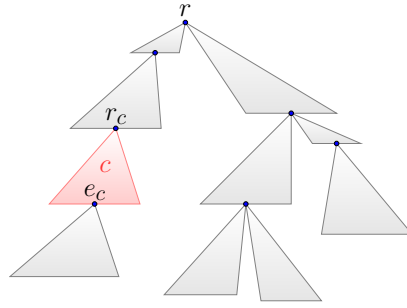
2 Preliminaries

Let T be a rooted tree (V, E) with root $r \in V$ and edge weights $w(u, v) \geq 0$ for all $(u, v) \in E$. The root r represents the *depot* of the tours. Let n denote the number of vertices in V . Let $V' \subseteq V$ denote the set of *terminals*, such that a *token* is placed on each terminal $v \in V'$. Let $k \in [1, n]$ be an integer *capacity* of the tours. The *cost* of a tour t , denoted by $\text{cost}(t)$, is the overall weight of the edges on that tour. We say that a tour *visits* a terminal $v \in V'$ if the tour picks up the token at v .²

► **Definition 8** (unit demand tree CVRP). *An instance of the unit demand version of the capacitated vehicle routing problem (CVRP) on trees consists of*

- an edge weighted tree $T = (V, E)$ with $n = |V|$ and with root $r \in V$ representing the depot,
- a set $V' \subseteq V$ of terminals,
- a positive integer tour capacity k such that $k \leq n$.

² Note that a tour might go through a terminal v without picking up the token at v .



■ **Figure 4** Decomposition into components. In this example, the tree is decomposed into four *leaf* components and six *internal* components. An internal component c has a *root* vertex r_c and an *exit* vertex e_c .

A *feasible solution* is a set of tours such that

- each tour starts and ends at r ,
- each tour visits at most k terminals,
- each terminal is visited by one tour.

The goal is to find a feasible solution such that the total cost of the tours is minimum.

Let OPT (resp. OPT_1 , OPT_2 , or OPT_3) denote an optimal (resp. near-optimal) solution to the unit demand CVRP, and let opt (resp. opt_1 , opt_2 , or opt_3) denote the value of that solution.

Without loss of generality, we assume that every vertex in the input tree T has exactly two children, and that the terminals are the same as the leaf vertices of the tree. Indeed, general instances can be reduced to instances with these properties by inserting edges of weight 0, removing leaf vertices that are not terminals, and slicing out internal vertices of degree two, see, e.g. [10] for details.

For any vertex $v \in V$, a *subtour at the vertex v* is a path that starts and ends at v and only visits vertices in the subtree rooted at v . The *demand* of a subtour is the number of terminals visited by that subtour. For each vertex $v \in V$, let $\text{dist}(v)$ denote the distance between v and the depot in the tree T . For technical reasons, we allow *dummy* terminals to be included in the solution at internal vertices of the tree.

Throughout the paper, we define several constants depending on ϵ : Γ (Lemma 9), α (Theorem 13), H_ϵ (Lemma 21), and β (Theorem 26). They satisfy the relation that $H_\epsilon \gg \Gamma \gg 1 \gg \epsilon \gg \alpha \gg \beta$.

Decomposition of the Tree into Components

The component decomposition (Lemma 9) is inspired by the cluster decomposition by Becker and Paul [10]. The proof of Lemma 9 is similar to arguments in [10]; we give its proof in the full version of the paper for completeness.

► **Lemma 9.** Let $\Gamma = \frac{12}{\epsilon}$. There is a polynomial time algorithm to compute a partition of the edges of the tree T into a set \mathcal{C} of components (see Figure 4), such that all of the following properties are satisfied:

- Every component $c \in \mathcal{C}$ is a connected subgraph of T ; the root vertex of the component c , denoted by r_c , is the vertex in c that is closest to the depot.
- We say that a component $c \in \mathcal{C}$ is a *leaf component* if all descendants of r_c in tree T are in c , and is an *internal component* otherwise. A leaf component c interacts with other components at vertex r_c only. An internal component c interacts with other components at two vertices only: at vertex r_c , and at another vertex, called the *exit vertex* of the component c , and denoted by e_c .

- Every component $c \in \mathcal{C}$ contains at most $2\Gamma \cdot k$ terminals. We say that a component is big if it contains at least $\Gamma \cdot k$ terminals. Each leaf component is big.
- If the number of components in \mathcal{C} is strictly greater than one, then we have: (1) there exists a map from all components to big components, such that the image of a component is among its descendants (including itself), and each big component has at most three pre-images; and (2) the number of components in \mathcal{C} is at most $3/\Gamma$ times the total demand in the tree T .

► **Remark 10.** The root and the exit vertices of components are a rough analog of *portals* used in approximation schemes for other problems: they are places where the dynamic program will gather and synthesize information about partial solutions before passing it on.

Compared with [10], the decomposition in Lemma 9 uses different parameters: the number of terminals inside a *leaf component* is $\Theta(k/\epsilon)$, whereas in [10] the number of terminals inside a *leaf cluster* is $\Theta(\epsilon \cdot k)$; the threshold demand to define *big components* is $\Theta(k/\epsilon)$, whereas in [10] the threshold demand to define *small clusters* is $\Theta(\epsilon^2 \cdot k)$.

► **Definition 11** (subtours in components and subtour types). *Let c be any component. A subtour in the component c is a path that starts and ends at the root r_c of the component, and such that every vertex on the path is in c . The type of a subtour is “passing” if c is an internal component and the exit vertex e_c belongs to that subtour; and is “ending” otherwise.*

A passing subtour in c is to be combined with a subtour at e_c . In a leaf component, there is no passing subtour.

► **Definition 12** (spine subtour). *For an internal component c , we define the spine subtour in the component c , denoted by spine_c , to be the connection (in both directions) between the root vertex r_c and the exit vertex e_c of the component c , without visiting any terminal.*

From the definition, a spine subtour in a component is also a passing subtour in that component.

Without loss of generality, we assume that any subtour in a component c either visits at least one terminal in c or is a spine subtour; that any tour traverses each edge of the tree at most twice (once in each direction); and that any tour contains at most one subtour in any component.³

3 Solutions Inside Components

In this section, we prove Theorem 13, which is a main novelty in this paper.

► **Theorem 13.** *Let $\alpha = \epsilon^{(\frac{1}{\epsilon}+1)}$. Consider the unit demand CVRP on the tree T with bounded distances. There exist dummy terminals and a solution OPT_1 visiting all of the real and the dummy terminals, such that all of the following holds:*

1. For each component c , there are at most $\frac{2\Gamma}{\alpha} + 1$ tours visiting terminals in c ;
2. For each component c and each tour t visiting terminals in c , the number of the terminals in c visited by t is at least $\alpha \cdot k$;
3. We have $\text{opt}_1 \leq (1 + \epsilon) \cdot \text{opt}$.

³ If a tour contains several subtours in a component c , we may combine those subtours into a single subtour in c without increasing the cost of the tour.

3.1 Construction of OPT_1

► **Definition 14** (large and small subtours). *We say that a subtour is large if its demand is at least $\alpha \cdot k$, and small otherwise.*

The construction of OPT_1 , starting from an optimal solution OPT , is in several steps.

Step 1: Detaching small subtours

Prune each tour of OPT so that it only visits the terminals that do *not* belong to a small subtour in any component, and is minimal. In other words, if a tour t in OPT contains a small ending subtour t_e in a component c , then we remove t_e from t ; and if a tour t in OPT contains a non-spine small passing subtour t_p in a component c , then we remove t_p from t , except for the spine subtour of c .

Let A denote the set of the resulting tours. Note that each tour in A is connected. The removed pieces of a non-spine small passing subtour t_p may be disconnected from one another.

The parts of OPT that have been pruned consist of the set \mathcal{E} , each element being a small ending subtours in a component, and of the set \mathcal{P} , each element being a group of pieces in a component obtained from a non-spine small passing subtour by removing the corresponding spine subtour. The *demand* of a group of pieces in \mathcal{P} is the number of terminals in all of the pieces in that group.

Step 2: Combining small subtours within components

- For each component c , repeatedly concatenate subtours in c from the set \mathcal{E} so that in the end, all resulting subtours in c from the set \mathcal{E} have demands between $\alpha \cdot k$ and $2\alpha \cdot k$ except for at most one subtour.
- For each component c , repeatedly merge groups in c from the set \mathcal{P} so that in the end, all resulting groups in c from the set \mathcal{P} have demands between $\alpha \cdot k$ and $2\alpha \cdot k$ except for at most one group.

Let \mathcal{E}' and \mathcal{P}' denote the corresponding sets after the modifications for all components c . Let $B = \mathcal{E}' \cup \mathcal{P}'$. An *element* of B is either a *subtour* or a *group of pieces* in a component c . For each component c , all elements of B in the component c have demands between $\alpha \cdot k$ and $2\alpha \cdot k$ except for at most two elements with smaller demands.

Step 3: Reassigning small subtours

Construct a bipartite graph with vertex sets A and B and with edge set E . Let a be any tour in A , and let a_0 denote the corresponding tour in OPT . Let b be any element in B . The set E contains an edge between a and b if and only if the element b contains terminals from a_0 ; the weight of the edge (a, b) is the number of terminals in both b and a_0 . By Lemma 1 from [10], there exists an assignment $f : B \rightarrow A$ such that each element $b \in B$ is assigned to a tour $a \in A$ with $(a, b) \in E$ and that, for each tour $a \in A$, the demand of a plus the overall demand of the elements $b \in B$ that are assigned to a is at most the demand of the corresponding tour a_0 plus the maximum demand of any element in B .

Let A_1 denote the set of *pseudo-tours* induced by the assignment f . Each pseudo-tour in A_1 is the union of a tour $a \in A$ and the elements $b \in B$ that are assigned to a .

Step 4: Correcting tour capacities

For each pseudo-tour a_1 in A_1 , if the demand of a_1 exceeds k , we repeatedly remove an element $b \in B$ from a_1 , until the demand of a_1 is at most k .

Let A_2 denote the resulting set of pseudo-tours. Every pseudo-tour in A_2 is a connected tour of demand at most k (Lemma 15). Let $B^* \subseteq B$ denote the set of the removed elements $b \in B$. The elements in B^* are represented by the small pieces in Figure 2a (Page 4).

Step 5: Creating additional tours

We connect the elements of B^* to the depot by creating additional tours as follows.

- Let Q denote the union of the spine subtours for all internal components. Q is represented by the green thick paths in Figure 2a. Let X denote a multi-subgraph of T that is the union of the elements in B^* and the edges in Q . Observe that each element in B^* is connected to the depot through edges in Q . Thus X is connected, and induces a traveling salesman tour t_{TSP} visiting all terminals in B^* . Without loss of generality, we assume that, for any component c , if t_{TSP} visits terminals in c , then those terminals belong to a single subpath p_c of t_{TSP} , such that p_c does not visit any terminal from other components.
- If the traveling salesman tour t_{TSP} is within the tour capacity, then let A_3 denote the set consisting of a single tour t_{TSP} . Otherwise, we apply the *iterated tour partitioning (ITP)* algorithm [20] on t_{TSP} : we partition t_{TSP} into segments with exactly k terminals each, except possibly the last segments containing less than k terminals. For each segment, we connect its endpoints to the depot so as to make a tour, see Figure 2b. Let A_3 denote the resulting set of tours.

Let $A_4 = A_2 \cup A_3$.

Step 6: Postprocessing

For each component c , we rearrange the small subtours in A_4 as follows.

- For each tour t in A_4 that contains a small subtour in c , letting t_c denote this small subtour, if t_c is an ending subtour, we remove t_c from t ; if t_c is a passing subtour, we remove t_c from t , except for the spine subtour in c . The total demand of all of the removed small subtours in c is at most k (Lemma 16).
- We create an additional tour t_c^* from the depot that connects all of the removed small subtours in c . If the demand of t_c^* is less than $\alpha \cdot k$, then we include dummy terminals at r_c into the tour t_c^* so that its demand is exactly k .

Let A_5 denote the resulting set of tours after removing small subtours from A_4 . Let A_6 denote the set of newly created tours $\{t_c^*\}_{c \in \mathcal{C}}$.

Finally, let $\text{OPT}_1 = A_5 \cup A_6$.

In Section 3.2, we show that OPT_1 is a feasible solution to the unit demand tree CVRP; in Section 3.3, we prove the three properties of OPT_1 in the claim of Theorem 13.

3.2 Feasibility of the Construction

► **Lemma 15.** *Every pseudo-tour in A_2 is a connected tour of demand at most k .*

Proof. Let a_2 be any pseudo-tour in A_2 . By the construction in Step 4, the demand of a_2 is at most k . It suffices to show that a_2 is a connected tour.

Observe that a_2 is the union of a tour $a \in A$ and some elements b_1, \dots, b_q from B , for $q \geq 0$. From the construction, any tour $a \in A$ is connected. Consider any b_i for $i \in [1, q]$. Observe that $f(b_i) = a$, so the edge (a, b_i) belongs to the bipartite graph (A, B) . If b_i is a

subtour at r_c for some component c , then r_c must belong to the tour a ; and if b_i is a group of pieces in some component c , then the spine subtour of c must belong to the tour a . So the union of a and b_i is connected. Thus $a_2 = a \cup b_1 \cup \dots \cup b_q$ is a connected tour. ◀

► **Lemma 16.** *In any component c , the total demand of all of the removed small subtours in c at the beginning of Step 6 is at most k .*

Proof. Let c be any component. The key is to show that the number of non-spine small subtours in c that are contained in tours in A_4 is at most 4. Since $A_4 = A_2 \cup A_3$, we analyze the number of non-spine small subtours in c that are contained in tours in A_2 and in A_3 , respectively.

The tours in A_2 contain at most two non-spine small subtours in c , since at most two elements of B in component c have demands less than $\alpha \cdot k$.

We claim that the tours in A_3 contain at most two non-spine small subtours in c . If A_3 consists of a single tour t_{TSP} , the claim follows trivially since any tour contains at most one subtour in c from our assumption. Next, we consider the case when A_3 is generated by the ITP algorithm. From our assumption, if t_{TSP} visits terminals in c , then those terminals belong to a single subpath p_c of t_{TSP} , such that p_c does not visit any terminal from other components. By applying the ITP algorithm on t_{TSP} , we obtain a collection of segments. All segments that intersect p_c visit exactly k terminals in c , except for possibly the first and the last of those segments visiting less than k terminals in c . Hence at most two non-spine small subtours in c among the tours in A_3 .

Therefore, the number of non-spine small subtours in c in the solution A_4 is at most 4. Since each small subtour has demand at most $\alpha \cdot k$, the total demand of the removed small subtours is at most $4 \cdot \alpha \cdot k < k$. ◀

3.3 Analysis of OPT_1

Let $c \in \mathcal{C}$ be any component. From Lemma 9, c contains at most $2\Gamma \cdot k$ real terminals. Each tour in A_5 visiting terminals in c visits at least $\alpha \cdot k$ real terminals in c , so there are at most $\frac{2\Gamma}{\alpha}$ tours in A_5 visiting terminals in c . There is a single tour in A_6 , the tour t_c^* , that visits terminals in c . Hence the first property of the claim. From the construction of t_c^* , the second property of the claim follows.

It remains to analyze the cost of OPT_1 . Compared with OPT , the extra cost in OPT_1 is due to Step 5 and Step 6 of the construction. This extra cost consists of the cost of the edges in Q (Step 5), the cost in the ITP algorithm to connect the endpoints of all segments to the depot (Step 5), and the cost of the postprocessing (Step 6), which we bound in Corollary 18 and Lemmas 19 and 20, respectively. Both Corollary 18 and Lemma 20 are based on Lemma 17.

► **Lemma 17.** *We have $\sum_{\text{component } c} \text{dist}(r_c) \leq \frac{\epsilon}{8} \cdot \text{opt}$.*

Proof. For any edge e in T , let u and v denote the two endpoints of e such that u is the parent of v . Let T_e denote the subtree of T rooted at v . Let n_e denote the number of terminals in T_e . From the lower bound in [21], we have

$$\text{opt} \geq \sum_{e \in T} 2 \cdot w(e) \cdot \frac{n_e}{k}.$$

Since each big component contains at least $\Gamma \cdot k$ terminals, we have

$$n_e \geq \sum_{\text{big component } c \subseteq T_e} \Gamma \cdot k.$$

Thus

$$\begin{aligned}
\text{opt} &\geq \sum_{e \in T} 2 \cdot w(e) \cdot \sum_{\substack{\text{big component } c \\ c \subseteq T_e}} \Gamma \\
&= \sum_{\text{big component } c} \Gamma \cdot \sum_{e \in T \text{ such that } c \subseteq T_e} 2 \cdot w(e) \\
&= \sum_{\text{big component } c} \Gamma \cdot 2 \cdot \text{dist}(r_c).
\end{aligned}$$

From Lemma 9, there exists a map from all components to big components such that the image of a component is among its descendants (including itself) and each big component has at most three pre-images. Thus

$$3 \cdot \sum_{\text{big component } c} \text{dist}(r_c) \geq \sum_{\text{component } c} \text{dist}(r_c).$$

Therefore,

$$\text{opt} \geq \frac{2 \cdot \Gamma}{3} \cdot \sum_{\text{component } c} \text{dist}(r_c).$$

The claim follows since $\Gamma = \frac{12}{\epsilon}$. ◀

► **Corollary 18.** *We have $\text{cost}(Q) \leq \frac{\epsilon}{4} \cdot \text{opt}$.*

Proof. Observe that every edge in Q belongs to the connection (in both directions) between the depot and the root of some component c . By Lemma 17, we have

$$\text{cost}(Q) \leq 2 \cdot \sum_{\text{component } c} \text{dist}(r_c) \leq \frac{\epsilon}{4} \cdot \text{opt}. \quad \blacktriangleleft$$

► **Lemma 19.** *Let Δ_1 denote the cost in the ITP algorithm to connect the endpoints of all segments to the depot in Step 5. We have $\Delta_1 \leq \frac{\epsilon}{4} \cdot \text{opt}$.*

Proof. Let n' denote the number of terminals in the tree T . First, we show that the number of terminals on t_{TSP} is at most $4\alpha \cdot n'$. Observe that the number of terminals on t_{TSP} is the overall removed demand in Step 4. Consider any pseudo-tour $a_1 \in A_1$ whose demand exceeds k . Let a_0 denote the corresponding tour in OPT. By the construction, the demand of a_1 is at most the demand of a_0 plus the maximum demand of any element in B . Since the demand of a_0 is at most k and the demand of any element in B is at most $2\alpha \cdot k$, the demand of a_1 is at most $k + 2\alpha \cdot k$. Let a_2 denote the corresponding tour after the correction of capacity in Step 4. Since any element in B has demand at most $2\alpha \cdot k$, the demand of a_2 is at least $k - 2\alpha \cdot k$. So the total removed demand from a_1 in Step 4 is at most $4\alpha \cdot k$. There are at most $\frac{n'}{k}$ pseudo-tours $a_1 \in A_1$ whose demands exceed k . Summing over all those pseudo-tours, the overall removed demand in Step 4 is at most $\frac{n'}{k} \cdot 4\alpha \cdot k = 4\alpha \cdot n'$. Hence the number of terminals on t_{TSP} is at most $4\alpha \cdot n'$.

If $4\alpha \cdot n' \leq k$, then t_{TSP} is within the tour capacity, so the ITP algorithm is not applied and $\Delta_1 = 0$. It remains to consider the case in which $4\alpha \cdot n' > k$. By the construction in the ITP algorithm, every segment visits exactly k terminals except possibly the last segment. Thus the number ℓ_{ITP} of resulting tours in the ITP algorithm is

$$\ell_{\text{ITP}} \leq \frac{4\alpha \cdot n'}{k} + 1. \quad (1)$$

95:14 A PTAS for Capacitated Vehicle Routing on Trees

Since $4\alpha \cdot n' > k$, we have $\ell_{\text{IPT}} < \frac{8\alpha \cdot n'}{k}$. Since $\Delta_1 \leq \ell_{\text{IPT}} \cdot 2 \cdot D_{\max}$ and using Definition 3 and the definition of α in the claim of Theorem 13, we have

$$\Delta_1 < \frac{8\alpha \cdot n'}{k} \cdot 2 \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{\epsilon}-1} \cdot D_{\min} < \frac{\epsilon}{2} \cdot \frac{n'}{k} \cdot D_{\min}.$$

On the other hand, the solution OPT consists of at least $\frac{n'}{k}$ tours, so $\text{opt} \geq \frac{n'}{k} \cdot D_{\min}$. Therefore, $\Delta_1 \leq \frac{\epsilon}{4} \cdot \text{opt}$. \blacktriangleleft

► **Lemma 20.** *Let Δ_2 denote the cost of the postprocessing (Step 6). Then $\Delta_2 \leq \frac{\epsilon}{2} \cdot \text{opt}$.*

Proof. For each leaf component c , the cost to connect the small subtours in c to the depot in Step 6 is at most $2 \cdot \text{dist}(r_c)$; and for each internal component c , the cost to connect the small subtours in c to the depot is at most $2 \cdot \text{dist}(r_c) + \text{cost}(\text{spine}_c)$. Summing over all components, we have

$$\Delta_2 \leq \sum_{\text{component } c} 2 \cdot \text{dist}(r_c) + \sum_{\text{internal component } c} \text{cost}(\text{spine}_c).$$

By Lemma 17, we have

$$\sum_{\text{component } c} 2 \cdot \text{dist}(r_c) \leq \frac{\epsilon}{4} \cdot \text{opt}$$

and

$$\sum_{\text{internal component } c} \text{cost}(\text{spine}_c) = \text{cost}(Q) \leq \frac{\epsilon}{4} \cdot \text{opt}.$$

Thus $\Delta_2 \leq \frac{\epsilon}{2} \cdot \text{opt}$. \blacktriangleleft

From Corollary 18 and Lemmas 19 and 20, we have $\text{opt}_1 \leq \text{opt} + \text{cost}(Q) + \Delta_1 + \Delta_2 \leq (1 + \epsilon) \cdot \text{opt}$. This completes the proof for Theorem 13.

4 Height Reduction

In this section, we transform the tree T into a tree \hat{T} so that \hat{T} has $O_\epsilon(1)$ levels of components, see Figure 1. We assume that the tree T has bounded distances. To begin with, we partition the components according to the distances from their roots to the depot.

► **Lemma 21.** *Let $\tilde{D} = \alpha \cdot \epsilon \cdot D_{\min}$. Let $H_\epsilon = \left(\frac{1}{\epsilon}\right)^{\frac{2}{\epsilon}+1}$. For each $i \in [1, H_\epsilon]$, let $\mathcal{C}_i \subseteq \mathcal{C}$ denote the set of components $c \in \mathcal{C}$ such that $\text{dist}(r_c) \in [(i-1) \cdot \tilde{D}, i \cdot \tilde{D})$. Then any component $c \in \mathcal{C}$ belongs to a set \mathcal{C}_i for some $i \in [1, H_\epsilon]$.*

Proof. Let $c \in \mathcal{C}$ be any component. We have

$$\text{dist}(r_c) \leq D_{\max} < \left(\frac{1}{\epsilon}\right)^{\frac{1}{\epsilon}-1} \cdot D_{\min} = H_\epsilon \cdot \tilde{D},$$

where the second inequality follows from Definition 3, and the equality follows from the definition of α in Theorem 13 and the definitions of \tilde{D} and H_ϵ . Thus there exists $i \in [1, H_\epsilon]$ such that $c \in \mathcal{C}_i$. \blacktriangleleft

► **Definition 22** (maximally connected sets and critical vertices). *We say that a set of components $\tilde{\mathcal{C}} \subseteq \mathcal{C}_i$ is maximally connected if the components in $\tilde{\mathcal{C}}$ are connected to each other and $\tilde{\mathcal{C}}$ is maximal within \mathcal{C}_i . For a maximally connected set of components $\tilde{\mathcal{C}} \subseteq \mathcal{C}_i$, we define the critical vertex of $\tilde{\mathcal{C}}$ to be the root vertex of the component $c \in \tilde{\mathcal{C}}$ that is closest to the depot.*

Figure 1 (Page 3) is an example with three levels of components: \mathcal{C}_1 , \mathcal{C}_2 , and \mathcal{C}_3 , indicated by different colors. There are four maximally connected sets of components. The critical vertices are represented by rectangular nodes.

■ **Algorithm 1** Construction of the tree \hat{T} (see Figure 1).

```

1: for each  $i \in [1, H_\epsilon]$  do
2:   for each maximally connected set of components  $\tilde{\mathcal{C}} \subseteq \mathcal{C}_i$  do
3:      $z \leftarrow$  critical vertex of  $\tilde{\mathcal{C}}$ 
4:     for each component  $c \in \tilde{\mathcal{C}}$  do
5:        $\delta \leftarrow$   $r_c$ -to- $z$  distance in  $T$ 
6:       Split the tree  $T$  at the root vertex  $r_c$  of the component  $c$    $\triangleright r_c$  is duplicated
7:       Add an edge between the root of the component  $c$  and  $z$  with weight  $\delta$ 
8:  $\hat{T} \leftarrow$  the resulting tree

```

Let \hat{T} be the tree constructed in Algorithm 1. We observe that Algorithm 1 is in polynomial time, and Fact 6 follows from the construction. We show in Theorem 23 that the optimal cost for \hat{T} is increased by an $O(\epsilon)$ fraction compared with the optimal cost for T .

► **Theorem 23.** *Consider the unit demand CVRP on the tree \hat{T} . There exist dummy terminals and a solution OPT_2 visiting all of the real and the dummy terminals, such that all of the following holds:*

1. *For each component c , there are at most $\frac{2\Gamma}{\alpha} + 1$ tours visiting terminals in c ;*
2. *For each component c and each tour t visiting terminals in c , the number of the terminals in c visited by t is at least $\alpha \cdot k$;*
3. *We have $\text{opt}_2 < (1 + 3\epsilon) \cdot \text{opt}$, where opt denotes the optimal cost for the unit demand CVRP on the tree T .*

In the rest of the section, we prove Theorem 23.

4.1 Construction of OPT_2

Consider any tour t in OPT_1 . Let U denote the set of terminals visited by t .⁴ We define the tour \hat{t} as the minimal tour in the tree \hat{T} that spans all terminals in U , see Figure 1. Let OPT_2 denote the set of the resulting tours on the tree \hat{T} constructed from every tour t in OPT_1 . Then OPT_2 is a feasible solution to the unit demand CVRP on \hat{T} .

4.2 Analysis of OPT_2

The first two properties in Theorem 23 follow from the construction and Theorem 13.

In the rest of the section, we analyze the cost of OPT_2 .

► **Lemma 24.** *Let t denote any tour in OPT_1 . Let \hat{t} denote the corresponding tour in OPT_2 . Then $\text{cost}(\hat{t}) \leq (1 + \epsilon) \cdot \text{cost}(t)$.*

⁴ We assume that t is a minimal tour in T spanning all terminals in U .

Proof. We follow the notation on U from Section 4.1. Let $\mathcal{C}(U)$ denote the set of components $c \in \mathcal{C}$ that contains a (possibly spine) subtour of \hat{t} . Observe that the cost of \hat{t} consists of the following two parts:

1. for each component $c \in \mathcal{C}(U)$, the cost of the subtour in c from the tour t ; we *charge* that cost to the subtour in t ;
2. for each component $c \in \mathcal{C}(U)$, the cost of the edge (r_c, z) , where z denotes the father vertex of r_c in \hat{T} . Note that z is a critical vertex on \hat{t} . We analyze that cost over all components $c \in \mathcal{C}(U)$ as follows.

Let $Z \subseteq V$ denote the set of critical vertices $z \in V$ on \hat{t} . For any critical vertex $z \in Z$, let $Y(z)$ denote the set of edges (z, v) in the tree \hat{T} such that v is a child of z and that the edge (z, v) belongs to the tour \hat{t} . The overall cost of the second part is the total cost of the edges in $Y(z)$ for all $z \in Z$.

Fix a critical vertex $z \in Z$. Let (z, v_1) denote the edge in $Y(z)$ such that $\text{dist}(v_1)$ is minimized, breaking ties arbitrarily. From the minimality of $\text{dist}(v_1)$, the z -to- v_1 path in T does not go through any component in $\mathcal{C}(U)$. From the construction, the cost of the edge (z, v_1) in \hat{T} equals the cost of the z -to- v_1 path in T . It is easy to see that the z -to- v_1 path in T belongs to the tour t . Indeed, tour \hat{t} traverses the edge (z, v_1) on its way to visit some terminals of U in the subtree rooted at v_1 . In order to visit the corresponding terminals in T , tour t must traverse the z -to- v_1 path. We *charge* the cost of the edge (z, v_1) in \hat{T} to the z -to- v_1 path in T . Next, we analyze the cost due to the other edges in $Y(z)$. Consider one such edge (z, v) . From the construction, there exists $i \in [1, H_\epsilon]$, such that both $\text{dist}(z)$ and $\text{dist}(v)$ belong to $[(i-1) \cdot \tilde{D}, i \cdot \tilde{D})$. Thus the cost of the z -to- v path in T equals $\text{dist}(v) - \text{dist}(z) < \tilde{D}$, so the extra cost in \hat{t} due to the edge (z, v) is at most $2 \cdot \tilde{D}$ (for both directions). Therefore, the extra cost in \hat{t} due to those $|Y(z)| - 1$ edges in $Y(z)$ is at most $2 \cdot \tilde{D} \cdot (|Y(z)| - 1)$.

Summing over all vertices $z \in Z$, and observing that all charges are to disjoint parts of t , we have

$$\text{cost}(\hat{t}) \leq \text{cost}(t) + 2 \cdot \tilde{D} \cdot \sum_{z \in Z} (|Y(z)| - 1). \quad (2)$$

It remains to bound $\sum_{z \in Z} (|Y(z)| - 1)$. The analysis uses the following basic fact in trees.

► **Fact 25.** *Let H be a tree with L leaves. For each vertex u in H , let $m(u)$ denote the number of children of u in H . Then*

$$\sum_{u \in H} (m(u) - 1) \leq L - 1.$$

We construct a tree H as follows. Starting from the tree spanning U in \hat{T} , we contract vertices in each component $c \in \mathcal{C}(U)$ into a single vertex; let H denote the resulting tree. It is easy to see that each leaf in H corresponds to a component $c \in \mathcal{C}(U)$ that contains at least one terminal in U (using the definition of $\mathcal{C}(U)$ and the fact that any descending component of c do not belong to $\mathcal{C}(U)$). From the second property of Theorem 23 (which follows from Theorem 13), terminals in U belong to at most $1/\alpha$ components. Thus, by Fact 25 we have

$$\sum_{z \in Z} (|Y(z)| - 1) \leq (1/\alpha) - 1.$$

Combined with Equation (2), we have

$$\text{cost}(\hat{t}) - \text{cost}(t) < 2 \cdot \tilde{D} \cdot (1/\alpha) = 2 \cdot \alpha \cdot \epsilon \cdot D_{\min} \cdot (1/\alpha) = 2\epsilon \cdot D_{\min},$$

using the definition of \tilde{D} in Lemma 21. Since $\text{cost}(t) \geq 2 \cdot D_{\min}$, the claim follows. ◀

Applying Lemma 24 on each tour t in OPT_1 and summing, we have $\text{opt}_2 \leq (1 + \epsilon) \cdot \text{opt}_1$. By Theorem 13, $\text{opt}_1 \leq (1 + \epsilon) \cdot \text{opt}$, thus $\text{opt}_2 \leq (1 + 3\epsilon) \cdot \text{opt}$. This completes the proof of Theorem 23.

5 Adaptive Rounding on the Subtour Demands

In this section, we prove Theorem 26. We use the adaptive rounding to show that, in a near-optimal solution, the demands of the subtours at any critical vertex are from a set of $O_\epsilon(1)$ values. This property enables us to later guess those values in polynomial time by a dynamic program.

► **Theorem 26.** *Let $\beta = \frac{1}{4} \cdot \epsilon^{(\frac{4}{\epsilon}+1)}$. Consider the unit demand CVRP on the tree \hat{T} . There exist dummy terminals and a solution OPT_3 visiting all of the real and the dummy terminals, such that all of the following holds:*

1. *For each component c , there are at most $\frac{2\Gamma}{\alpha} + 1$ tours visiting terminals in c ;*
2. *For each critical vertex z , there exist $\frac{1}{\beta}$ integer values in $[\alpha \cdot k, k]$ such that the demands of the subtours at the children of z are among these values;*
3. *We have $\text{opt}_3 < (1 + 4\epsilon) \cdot \text{opt}$, where opt denotes the optimal cost for the unit demand CVRP on the tree T .*

5.1 Construction of OPT_3

We construct the solution OPT_3 by modifying the solution OPT_2 .

Let $I \subseteq V$ denote the set of vertices $v \in V$ that is either the root of a component or a critical vertex. Consider any vertex $v \in I$ in the bottom up order. Let $\text{OPT}_2(v)$ denote the set of subtours at v in OPT_2 . We construct a set $A(v)$ of subtours at v satisfying the following invariants:

- the subtours in $A(v)$ have a one-to-one correspondence with the subtours in $\text{OPT}_2(v)$; and
- the demand of each subtour of $A(v)$ is at most that of the corresponding subtour in $\text{OPT}_2(v)$.

The construction of $A(v)$ is according to one of the following three cases on v .

Case 1: v is the root vertex r_c of a leaf component c in \hat{T}

Let $A(v) = \text{OPT}_2(v)$.

Case 2: v is the root vertex r_c of an internal component c in \hat{T}

For each subtour $a \in \text{OPT}_2(v)$, if a contains a subtour at the exit vertex e_c of component c , letting t denote this subtour and t' denote the subtour in $A(e_c)$ corresponding to t , we replace the subtour t in a by the subtour t' . Let $A(v)$ be the resulting set of subtours at v .

Case 3: v is a critical vertex in \hat{T}

We apply the technique of the *adaptive rounding*, previously used by Jayaprakash and Salavatipour [22] in their design of a QPTAS the tree CVRP. The idea is to *round up* the demands of the subtours at the children of v so that the resulting demands are among $\frac{1}{\beta}$ values.

Let r_1, \dots, r_m be the children of v in \hat{T} . For each subtour $a \in \text{OPT}_2(v)$ and for each $i \in [1, m]$, if a contains a subtour at r_i , letting t denote this subtour and t' denote the subtour

in $A(r_i)$ corresponding to t , we replace t in a by t' . Let $A_1(v)$ denote the resulting set of subtours at v .

Let W_v denote the set of the subtours at the children of v in $A_1(v)$, i.e., $W_v = A(r_1) \cup \dots \cup A(r_m)$. If $|W_v| \leq \frac{1}{\beta}$, let $A(v) = A_1(v)$. In the following, we consider the non-trivial case when $|W_v| > \frac{1}{\beta}$. We sort the subtours in W_v in non-decreasing order of their demands, and partition these subtours into $\frac{1}{\beta}$ groups of equal cardinality.⁵ We *round* the demands of the subtours in each group to the maximum demand in that group. The demand of a subtour is increased to the rounded value by adding *dummy* terminals at the children of v . We rearrange the subtours in W_v as follows.

- Each subtour $t \in W_v$ in the last group is discarded, i.e., detached from the subtour in $A_1(v)$ to which it belongs.
- Each subtour $t \in W_v$ in other groups is associated in a one-to-one manner to a subtour $t' \in W_v$ in the next group. Letting a (resp. a') denote the subtour in $A_1(v)$ to which t (resp. t') belongs, we detach t from a and reattach t to a' .

Let $A(v)$ be the set of the resulting subtours at v after the rearrangement for all $t \in W_v$.

For each subtour t that is discarded in the construction, we complete t into a separate tour by adding the connection (in both directions) to the depot. Let B denote the set of these newly created tours. Let $\text{OPT}_3 = A(r) \cup B$.

It is easy to see that OPT_3 is a feasible solution to the unit demand CVRP, i.e., each tour in OPT_3 is connected and visits at most k terminals, and each terminal is covered by some tour in OPT_3 .

5.2 Analysis of OPT_3

From the construction, in any component $c \in \mathcal{C}$, the non-spine subtours in OPT_3 are the same as those in OPT_2 . From Theorem 23, we obtain the first property in Theorem 26, and in addition, each subtour at a child of a critical vertex in OPT_2 has demand at least $\alpha \cdot k$. The second property of the claim follows from the construction of OPT_3 .

It remains to analyze the cost of OPT_3 . Let $\Delta = \text{opt}_3 - \text{opt}_2$. Observe that Δ is due to adding connections to the depot to create the tours in the set B .

Fix any $i \in [1, H_\epsilon]$. Let $Z \subseteq V$ denote the set of vertices $v \in V$ such that v is the critical vertex of a maximally connected component $\tilde{\mathcal{C}} \subseteq \mathcal{C}_i$. For any $v \in Z$, we analyze the number of discarded subtours in the set W_v defined in Section 5.1. If $|W_v| \leq \frac{1}{\beta}$, there is no discarded subtour in W_v ; if $|W_v| > \frac{1}{\beta}$, the number of discarded subtours in W_v is $\lceil \beta \cdot |W_v| \rceil < \beta \cdot |W_v| + 1 < 2\beta \cdot |W_v|$. Let W denote the disjoint union of W_v for all vertices $v \in Z$. Thus W contains at most $2\beta \cdot |W|$ discarded subtours. Let Δ_i denote the cost to connect the discarded subtours in W to the depot. We have

$$\Delta_i \leq 2\beta \cdot |W| \cdot 2 \cdot D_{\max} < \frac{1}{2} \cdot \epsilon^{\left(\frac{4}{\epsilon} + 1\right)} \cdot |W| \cdot 2 \cdot \left(\frac{1}{\epsilon}\right)^{\frac{1}{\epsilon} - 1} \cdot D_{\min} = \frac{\epsilon}{H_\epsilon} \cdot \alpha \cdot |W| \cdot D_{\min}, \quad (3)$$

where the second inequality follows from the definition of β (Theorem 26) and Definition 3, and the equality follows from the definitions of α (Theorem 13) and of H_ϵ (Lemma 21). From the second property of the claim, each subtour in W has demand at least $\alpha \cdot k$, so there are at least $\alpha \cdot |W|$ tours in OPT_3 . Any tour in OPT_3 has cost at least $2 \cdot D_{\min}$, so we have

$$\text{opt}_3 \geq 2 \cdot \alpha \cdot |W| \cdot D_{\min}. \quad (4)$$

⁵ We add empty subtours to the first groups if needed in order to achieve equal cardinality among all groups.

From Equations (3) and (4), we have

$$\Delta_i < \frac{\epsilon}{2 \cdot H_\epsilon} \cdot \text{opt}_3.$$

Summing over all integers $i \in [1, H_\epsilon]$, we have $\Delta = \sum_i \Delta_i \leq \frac{\epsilon}{2} \cdot \text{opt}_3$. Thus

$$\text{opt}_3 \leq \frac{2}{2 - \epsilon} \cdot \text{opt}_2.$$

By Theorem 23, $\text{opt}_2 \leq (1 + 3\epsilon) \cdot \text{opt}$. Therefore, $\text{opt}_3 \leq (1 + 4\epsilon) \cdot \text{opt}$. This completes the proof of Theorem 26.

References

- 1 Anna Adamaszek, Artur Czumaj, and Andrzej Lingas. PTAS for k -tour cover problem on the plane for moderately large values of k . *International Journal of Foundations of Computer Science*, 21(06):893–904, 2010.
- 2 Kemal Altinkemer and Bezalel Gavish. Heuristics for delivery problems with constant error guarantees. *Transportation Science*, 24(4):294–297, 1990.
- 3 S. P. Anbuudayasankar, K. Ganesh, and Sanjay Mohapatra. *Models for practical routing problems in logistics*. Springer, 2016.
- 4 Tetsuo Asano, Naoki Katoh, and Kazuhiro Kawashima. A new approximation algorithm for the capacitated vehicle routing problem on a tree. *Journal of Combinatorial Optimization*, 5(2):213–231, 2001.
- 5 Tetsuo Asano, Naoki Katoh, Hisao Tamaki, and Takeshi Tokuyama. Covering points in the plane by k -tours: towards a polynomial time approximation scheme for general k . In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 275–283, 1997.
- 6 Amariah Becker. A tight $4/3$ approximation for capacitated vehicle routing in trees. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 116 of *LIPICs*, pages 3:1–3:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 7 Amariah Becker, Philip N. Klein, and David Saulpic. A quasi-polynomial-time approximation scheme for vehicle routing on planar and bounded-genus graphs. In *25th Annual European Symposium on Algorithms (ESA 2017)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
- 8 Amariah Becker, Philip N. Klein, and David Saulpic. Polynomial-time approximation schemes for k -center, k -median, and capacitated vehicle routing in bounded highway dimension. In *26th Annual European Symposium on Algorithms (ESA 2018)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.
- 9 Amariah Becker, Philip N. Klein, and Aaron Schild. A PTAS for bounded-capacity vehicle routing in planar graphs. In *Workshop on Algorithms and Data Structures*, pages 99–111. Springer, 2019.
- 10 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.
- 11 Jannis Blauth, Vera Traub, and Jens Vygen. Improving the approximation ratio for capacitated vehicle routing. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 1–14. Springer, 2021.
- 12 Agustín Bompadre, Moshe Dror, and James B. Orlin. Improved bounds for vehicle routing solutions. *Discrete Optimization*, 3(4):299–316, 2006.
- 13 Vincent Cohen-Addad, Arnold Filtser, Philip N. Klein, and Hung Le. On light spanners, low-treewidth embeddings and efficient traversing in minor-free graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 589–600. IEEE, 2020.

- 14 Teodor G. Crainic and Gilbert Laporte. *Fleet management and logistics*. Springer Science & Business Media, 2012.
- 15 George B. Dantzig and John H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.
- 16 Aparna Das and Claire Mathieu. A quasipolynomial time approximation scheme for Euclidean capacitated vehicle routing. *Algorithmica*, 73(1):115–142, 2015.
- 17 Zachary Friggstad, Ramin Mousavi, Mirmahdi Rahgoshay, and Mohammad R. Salavatipour. Improved approximations for CVRP with unsplittable demands. *arXiv preprint*, 2021. [arXiv:2111.08138](https://arxiv.org/abs/2111.08138).
- 18 Bruce Golden, S. Raghavan, and Edward Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*. Springer, 2008.
- 19 Bruce L. Golden and Richard T. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.
- 20 Mordecai Haimovich and Alexander H. G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, 1985.
- 21 Shin-ya Hamaguchi and Naoki Katoh. A capacitated vehicle routing problem on a tree. In *International Symposium on Algorithms and Computation*, pages 399–407. Springer, 1998.
- 22 Aditya Jayaprakash and Mohammad R. Salavatipour. Approximation schemes for capacitated vehicle routing on graphs of bounded treewidth, bounded doubling, or highway dimension. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 877–893, 2022.
- 23 Michael Khachay and Roman Dubinin. PTAS for the Euclidean capacitated vehicle routing problem in \mathbb{R}^d . In *International Conference on Discrete Optimization and Operations Research*, pages 193–205. Springer, 2016.
- 24 Martine Labbé, Gilbert Laporte, and Hélène Mercure. Capacitated vehicle routing on trees. *Operations Research*, 39(4):616–622, 1991.
- 25 Claire Mathieu and Hang Zhou. A tight $(1.5 + \epsilon)$ -approximation for unsplittable capacitated vehicle routing on trees, 2022. [arXiv:2202.05691](https://arxiv.org/abs/2202.05691).
- 26 Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2002.