Parameterized Complexity of Vehicle Routing

Michelle Döring
Hasso Plattner Institute,
University of Potsdam, Germany

Jan Fehse
Hasso Plattner Institute,
University of Potsdam, Germany

Tobias Friedrich
Hasso Plattner Institute,
University of Potsdam, Germany

Paula Marten
Hasso Plattner Institute,
University of Potsdam, Germany

Niklas Mohrin
Hasso Plattner Institute,
University of Potsdam, Germany

Kirill Simonov
Department of Informatics,
University of Bergen, Norway

Farehe Soheil
Hasso Plattner Institute,
University of Potsdam, Germany

Jakob Timm
Hasso Plattner Institute,
University of Potsdam, Germany

Shaily Verma
Hasso Plattner Institute,
University of Potsdam, Germany

Abstract

The Vehicle Routing Problem (VRP) is a popular generalization of the Traveling Salesperson Problem. Instead of one salesperson traversing the entire weighted, undirected graph G, there are k vehicles available to jointly cover the set of clients $C \subseteq V(G)$. Every vehicle must start at one of the depot vertices $D \subseteq V(G)$ and return to its start. Capacitated Vehicle Routing (CVRP) additionally restricts the route of each vehicle by limiting the number of clients it can cover, the distance it can travel, or both.

In this work, we study the complexity of VRP and the three variants of CVRP for several parameterizations, in particular focusing on the treewidth of G. We present an FPT algorithm for VRP parameterized by treewidth. For CVRP, we prove paraNP- and W[$\cdot$]-hardness for various parameterizations, including treewidth, thereby rendering the existence of FPT algorithms unlikely. In turn, we provide an XP algorithm for CVRP when parameterized by both treewidth and the vehicle capacity.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Fixed parameter tractability

Keywords and phrases Vehicle Routing Problem, Treewidth, Parameterized Complexity

Digital Object Identifier 10.4230/LIPIcs.IPEC.2025.10

Related Version *Full Version*: `https://arxiv.org/abs/2509.10361` [8]

Funding *Michelle Döring*: German Federal Ministry for Education and Research (BMBF) through the project "KI Servicezentrum Berlin Brandenburg" (01IS22092).

1 Introduction

Optimizing logistics has been an important driver of the theory of computing since its very dawn, and, with the ever-growing digitalization and decentralization of services, it remains a productive direction to this day. One of the canonical challenges in the area, the Vehicle Routing Problem (VRP), introduced by George Dantzig and John Ramser in 1959 [6], can be informally summarized as follows: "What is the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers?"

To model this question, it is natural to use a weighted graph to represent the relevant locations and the travel costs between them, with particular vertices marked as the depots, where the vehicles and the goods are originally located, and with some other vertices marked as the clients that await the delivery. Formally, we consider VRP to be the following computational problem: Given an edge-weighted undirected graph (G, w), subsets $C, D \subseteq V(G)$, and integers k, r, determine whether there exists a collection of at most k closed walks in G, such that each walk starts and ends at a vertex of D, the walks cover all vertices in C, and their total weight is at most r.

Defined in this way, VRP also becomes interesting from the theoretical point of view, as it generalizes several important NP-hard problems. Notably, the famous Traveling Salesperson Problem is a special case of VRP with one vehicle and every vertex being a client, i.e., $k = 1$, $C = D = V(G)$. On the other hand, since VRP allows for multiple vehicles, it also captures packing and covering problems on graphs. For example, the Maximum Cycle Cover problem, where the task is to find a collection of k vertex-disjoint cycles in the given graph G that covers all vertices [5], is modeled by VRP with $C = D = V(G)$ and $r = |V(G)|$.

Based on the immediate hardness of the problem in the classical sense, practical solvers for vehicle routing are mostly based on heuristics, which provide guarantees on neither the quality of the solution nor on the running time. We refer to surveys by Braekers et al. [4] and by Mor and Speranza [11] for an in-depth introduction to the vast area of practical research on solving the vehicle routing problem, as well as for listing various variants of the vehicle routing problem that received practical interest. Stemming from numerous research works and the efficiency of the developed solvers, the vehicle routing problem itself sees applications to other domains, for example, routing traffic in computer networks [1].

On the other hand, theoretical results regarding VRP have been much scarcer, especially with respect to solving the problem exactly. The immediate NP-hardness of the problem together with the diversity of explicit (such as number of vehicles and tour lengths) and implicit parameters (such as treewidth that may be small in certain applications [1]) makes VRP a natural target for parameterized complexity. However, to the best of our knowledge, this avenue was largely unexplored until now, although TSP itself [3, 7] and other related problems such as cycle packing [2] are well-studied in the area.

Our contribution. In this work, we aim to close this gap and initiate the parameterized complexity study of VRP and its variants. First, based on the different variants of vehicle routing found in the literature [4, 11], we define a general vehicle routing setting GVRS that provides a unified interface to the individual problems. We then focus on two important special cases: the VRP problem, as defined above, and its capacitated variant, where each route is additionally subject to a size constraint. Specifically, in LoadCVRP, the input also contains the parameter ℓ, and each route in the solution can only serve at most ℓ clients. This is motivated by real-life constraints, such as the limited cargo space of vehicles and the limited working hours of drivers. On the other hand, the capacity constraint allows to encapsulate even more fundamental theoretical problems within the vehicle routing framework, such as Triangle Packing, corresponding to $\ell = 3$, and ℓ-Cycle Packing. Similarly, we define GasCVRP, where there is a limit on the edge-weight of each route, and LoadGasCVRP, where both edge-weight and number of clients are limited. The problem classification is covered in detail in Section 2.1.

We then proceed to investigate the parameterized complexity of these problems with respect to the parameters such as the number of vehicles k, the number of depots $|D|$, the number of clients $|C|$, the total weight of the solution r, the treewidth of the graph tw, and the maximum load ℓ in case of LoadCVRP. Our findings are summarized in Table 1.

Table 1 Overview of our results for Vehicle Routing variants. We omit the results for GasCVRP and LoadGasCVRP that are analogous to the listed results for LoadCVRP.

VRP Variant	Parameter	Complexity	Reference
VRP	$\lvert D\rvert + k$	paraNP-hard	Theorem 3.1 (MetricTSP)
	$\lvert C\rvert$	FPT	Theorem 3.2
	r	FPT	Corollary 3.3 ($r \geq \lvert C\rvert$)
	tw	FPT	Theorem 3.9
LoadCVRP	ℓ	paraNP-hard	Theorem 4.12 (TrianglePacking)
	$\lvert C\rvert$	FPT	Theorem 4.1
	r	FPT if 0-edges disallowed	Corollary 4.3 ($r \geq \lvert C\rvert$)
	$k + \ell$	FPT	Theorem 4.4 ($k \cdot \ell \geq \lvert C\rvert$)
	$\mathrm{tw} + \lvert D\rvert$	paraNP-hard	Theorem 4.8 (BinPacking)
	$\mathrm{tw} + k + \lvert D\rvert$	W[1]-hard	Theorem 4.8 (BinPacking)
	$\mathrm{tw} + \ell$	XP	Corollary 4.25
LoadGasCVRP	$\mathrm{tw} + \ell + \lvert D\rvert$	paraNP-hard	Theorem 4.15 (N3DMatching)
	$\mathrm{tw} + g + \lvert D\rvert$	paraNP-hard	Corollary 4.16 (N3DMatching)

We start in Section 3 with the uncapacitated case, i.e., the VRP problem. As this problem presents a generalization of TSP, we can rule out tractability even for a constant number of vehicles and depots in Theorem 3.1. In Theorem 3.2 we also quickly conclude that the number of clients is a sufficient parameter for an FPT algorithm. This result directly implies that deciding whether a routing of weight at most r exists can be done in FPT-time in r.

The main result of Section 3 deals with structural properties of our network, namely its treewidth. While there are many known FPT algorithms parameterized by treewidth for various graph problems, the VRP problem presents a unique combination of several challenging aspects. The solution is composed of arbitrarily many individual objects, i.e., walks, which may use edges and vertices multiple times and can have arbitrary length. Neither the number of walks in the solution, nor the length of the individual walk, nor the multiplicity of an edge or a vertex in a walk is bound by the treewidth. The closest known starting point on the approach to VRP is the FPT algorithm by Schierreich and Suchý [14] for the WAYPOINT ROUTING PROBLEM parameterized by treewidth, which is effectively a single-vehicle case of VRP. While resolving some of the challenges outlined above, their algorithm does not readily allow for the incorporation of the partitioning aspect as well, i.e., to deal with multiple vehicles, over which the clients need to be partitioned. We generalize this approach further and obtain an FPT algorithm for VRP parameterized by treewidth (Theorem 3.9). In fact, the result we obtain holds for the more general Edge-Capacitated Vehicle Routing Problem (EVRP), which additionally allows to specify arbitrary capacities on the edges, so that each edge can be used by the solution at most the specified number of times. One of the key ingredients behind the algorithm is the combinatorial observation that allows us to restrict the number of times an edge can be used by the solution. Based on this, we introduce an equivalent reformulation of VRP that only considers Eulerian submultigraphs. This reformulation can, in turn, be solved efficiently via dynamic programming over the tree decomposition. Unfortunately, the operations in dynamic programming required for dealing with multiple walks extend beyond the established single-exponential-time machinery developed by Bodlaender et al. [3]. The running time that our algorithm achieves is thus of the form $2^{\mathcal{O}(\mathrm{tw}\log\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$; improving this to purely single-exponential running time remains a challenging open question. Nevertheless, this result completes the classification with respect to the outlined parameters.

We then focus on capacitated vehicle routing in Section 4. Recall that LoadCVRP, GasCVRP, and LoadGasCVRP result from adding to VRP the restriction on the maximum number of clients per vehicle ℓ, the maximum length of a tour g, or both simultaneously. As our work shows, these three versions largely behave in the same way with respect to the parameters in question; therefore we mainly focus on LoadCVRP in order to introduce our results.

The additional constraints allow us to model different packing problems as capacitated vehicle routing problems, and by showing such inclusions, we derive several intractability results. In Section 4.2, we show that LoadCVRP is NP-hard even for maximum load $\ell = 3$, by providing a reduction from TrianglePacking. More surprisingly, by reduction from BinPacking, we can also show that treewidth alone is not enough for fixed-parameter tractability in the presence of the capacity constraints, even when combined with the number of vehicles k and the number of depot vertices $|D|$.

Based on this hardness result, we move on to consider treewidth in combination with the capacity parameters. We show complete intractability for LoadGasCVRP, even when combining treewidth, the maximum load, and the number of depots as a parameter, using N3DMatching in Section 4.3. However, XP tractability can be achieved by combining treewidth and the capacity parameter in LoadCVRP and GasCVRP, as well as combining both capacity parameters with treewidth in LoadGasCVRP. These algorithms are the main results of Section 4. To obtain these, we first introduce a generalization of the BinPacking problem that acts as an interface to the capacitated vehicle routing problems, and prove tractability of this problem for a combination of parameters. Finally, in Section 4.4 we present a dynamic programming algorithm that solves LoadGasCVRP by modeling the optimal combination of partial solutions as instances of generalized BinPacking.

The proofs of some theorems are omitted and can be found in the full version of this paper [8]. These theorems are marked with ($\star$).

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers, including zero. For $n \in \mathbb{N}$, we abbreviate $[n] = \{1, 2, \ldots, n\}$ and $[n]_0 = \{0\} \cup [n]$. A *multiset* is an unordered collection of elements of some universe U, where each element can occur multiple times. Formally, we model multisets as functions mapping each possible element to the number of occurrences. Alternatively, we use set notations with modified brackets as in these examples: $[\![1, 2, 2, 3]\!]$ (the multiset containing 1 and 3 once, and 2 twice), $[\![\lfloor n/2 \rfloor \mid n \in \mathbb{N}]\!]$ (the multiset containing each natural number twice). The cardinality of a multiset A is denoted as $|A| = \sum_{u \in U} A(u)$. For some sub-universe $U' \subseteq U$, we denote the restriction of A to U' as $A_{\downarrow U'}$. For two multisets A, B over the same universe U, we denote by $A + B$ the multiset union defined by $(A+B)(u) = A(u) + B(u)$ for every $u \in U$. For $n \in \mathbb{N}$, we denote by $n \cdot A$ the multiset obtained by uniting n copies of A. Let G be an undirected (multi-)graph. By convention, we usually abbreviate $n = |V(G)|$. A *walk* p in a graph G is defined as a sequence of vertices and edges $(v_1, e_1, v_2, e_2, \ldots, e_{\ell-1}, v_\ell)$, where the sequence begins at a vertex and traverses edges of the graph, allowing for the repetition of both vertices and edges. The *length* of a walk is given by the number of edges it contains, counted with multiplicity. For simplicity, we may represent a walk of length $\ell - 1$ by the sequence of its vertices, i.e., $(v_1, v_2, \ldots, v_\ell)$. A walk is *closed* if the first and last visited vertices are equal, otherwise it is *open*.

A (multi-)graph G is called *Eulerian* if all vertices in G have even degree. Equivalently, G is Eulerian when it is the (multiset-)union of closed walks. Note that, unlike the classic (connected) notion of Eulerian graphs, this definition does not require G to be connected. Given an edge $e \in E(G)$ and number $k \in \mathbb{N}$, we denote by $G - k \cdot e$ the multigraph G with k occurrences of e removed.

Let $w\colon E(G) \to \mathbb{N}$ be a weight function for the edges of G. For a walk $p = (v_1, v_2, \dots, v_\ell)$ in G we define the weight of p as $w(p) = \sum_{i=1}^{\ell-1} w(v_i, v_{i+1})$. For a submultigraph $H \subseteq G$ we define the weight of H as $w(H) = \sum_{e \in E(H)} w(e)$, where parallel edges in H should appear separately in the sum, that is, the weight of H accounts for the multiplicity of the edges.

Finally, we recall the definitions of tree decompositions, treewidth, and nice tree decompositions, which we will use for our treewidth-based algorithms.

▶ **Definition 2.1** (Tree Decomposition, [13, 5]). A *tree decomposition* of an undirected graph G is a pair $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ consisting of a tree T and for each vertex t of T a set of vertices $X_t \subseteq V(G)$, so that
- $\bigcup_{t \in V(T)} X_t = V(G)$,
- for all $\{u, v\} \in E(G)$, there is $t \in V(T)$ such that $u, v \in X_t$, and
- for all $v \in V(G)$, the vertices $t \in V(T)$ with $v \in X_t$ form a connected subtree of T.

The sets X_t are called *bags*. We abbreviate $t \in V(T)$ as $t \in \mathcal{T}$. The *width* of $\mathcal{T}$ is defined as $\operatorname{width}(\mathcal{T}) = \max_{t \in \mathcal{T}} |X_t| - 1$ and the *treewidth* of G is the minimum width of any tree decomposition of G. ⌟

▶ **Definition 2.2** (Nice Tree Decomposition, [5]). A tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ is called *nice* if T is rooted at a vertex $r \in V(T)$ with $X_r = \emptyset$ and all vertices $t \in V(T)$ are of one of the following kinds:
- Leaf Node: t has no children and $X_t = \emptyset$.
- Introduce Vertex Node: t has exactly one child t' and there is a vertex $v \notin X_{t'}$ such that $X_t = X_{t'} \cup \{v\}$.
- Introduce Edge Node: t has exactly one child t', $X_t = X_{t'}$, and t is labeled with an edge $\{u, v\} \in E(G)$ with $u, v \in X_t$.
- Forget Node: t has exactly one child t' and there is a vertex $v \in X_{t'}$ such that $X_t = X_{t'} \setminus \{v\}$.
- Join Node: t has exactly two children t_1, t_2 and $X_t = X_{t_1} = X_{t_2}$.

For each edge $e \in E(G)$, there must be exactly one Introduce Edge Node labeled with e. If G is a multigraph, each copy of an edge is introduced separately.

For $t \in \mathcal{T}$, let S be the set of all descendants of t in T (including t itself). We denote the set of all introduced vertices up until t as $V_t^{\downarrow} = \bigcup_{t' \in S} X_{t'}$. Similarly, we define $E_t^{\downarrow}$ to be the (multi-)set of all introduced edges up until t and $G_t^{\downarrow} = (V_t^{\downarrow}, E_t^{\downarrow})$. ⌟

2.1 Classifying Vehicle Routing Problems

The literature deals with many variants of vehicle routing problems [4, 11]. To incorporate those in a single notion, we introduce the following *Generalized Vehicle Routing Setting.*

▶ **Definition 2.3** (Generalized Vehicle Routing Setting). An instance of the *Generalized Vehicle Routing Setting* (GVRS) consists of a graph G with edge weights $w\colon E(G) \to \mathbb{N}$, a nonempty set of *depots* $D \subseteq V(G)$, a set of *clients* $C \subseteq V(G)$, an integer $k \in \mathbb{N}$ representing the number of vehicles, and a predicate Γ modeling additional constraints. We call a set of k' walks $R = \{p_1, p_2, \dots, p_{k'}\}$ together with an assignment of vehicles $A\colon C \to [k']$ a *routing* for (G, w, D, C, k, Γ) if

1. $k' \leq k$ (the routing uses at most k vehicles),
2. $\forall p = (v_1, v_2, \ldots, v_\ell) \in R : v_1 \in D$ (every walk starts in one of the depots),
3. $\forall c \in C : c \in p_{A(c)}$ (every client is visited by its assigned vehicle), and
4. $\Gamma(R, A)$ (the additional constraints are satisfied).

The weight of a routing R is defined as $w(R) = \sum_{p \in R} w(p)$.

A problem instance may ask whether a feasible routing exists or whether there exists a routing R with $w(R) \leq r$, for some $r \in \mathbb{N}$. A routing of minimum weight is called *optimal.* ⌟

This paper studies four vehicle routing problems, denoted here as VRP, LoadCVRP, GasCVRP, and LoadGasCVRP for undirected graphs. The problem definitions will contain the additional constraint that each vehicle must return to the depot at which it started. More formally, we will use

$$\Gamma_{\text{closed}}(R, A) \iff \forall p \in R : p \text{ is closed.}$$

▶ **Definition 2.4** (VRP)**.** The *(Uncapacitated) Vehicle Routing Problem* (VRP) is a GVRS with an additional parameter $r \in \mathbb{N}$. It asks whether there exists a routing (R, A) with $w(R) \leq r$, subject to $\Gamma = \Gamma_{\text{closed}}$. ⌟

In Section 3, we obtain the main result for VRP, a parameterized algorithm for instances with bounded treewidth, by considering EVRP, a slightly generalized version of VRP.

▶ **Definition 2.5** (EVRP)**.** The *Edge-Capacitated Vehicle Routing Problem* (EVRP) is a GVRS with additional parameters $r \in \mathbb{N}$ and $\kappa\colon E(G) \to \mathbb{N}$. It asks whether there exists a routing (R, A) with $w(R) \leq r$, subject to $\Gamma = \Gamma_{\text{closed}} \wedge \Gamma_\kappa$, where

$$\Gamma_\kappa(R, A) \iff \forall e \in E : e \text{ is used at most } \kappa(e) \text{ times in } R.$$ ⌟

The literature on vehicle routing problems often considers *capacitated* variants [4, 11], usually denoted as CVRP. In contrast to EVRP, these capacities are placed on the vehicles individually. However, the term CVRP can refer to different kinds of capacities: *load* and *gas* capacities. In this paper, we consider both interpretations of CVRP.

In LoadCVRP, each client c has a demand for $\Lambda(c)$ units of some good, which must be delivered by its assigned vehicle. The vehicles, in turn, have a capacity ℓ on the *load* they can carry on their trip. Notably, this capacity ℓ is equal for all vehicles.

▶ **Definition 2.6** (LoadCVRP)**.** The *Load-Capacitated Vehicle Routing Problem* (LoadCVRP) is a GVRS with additional parameters $r \in \mathbb{N}$, $\ell \in \mathbb{N}$, and $\Lambda\colon C \to \mathbb{N}^+$. It asks whether there exists a routing (R, A) with $w(R) \leq r$, subject to $\Gamma = \Gamma_{\text{closed}} \wedge \Gamma_{\ell,\Lambda}$, where

$$\Gamma_{\ell,\Lambda}(R, A) \iff \forall i \in [|R|] : \sum_{c \in A^{-1}(i)} \Lambda(c) \leq \ell.$$ ⌟

A similar restriction is given in GasCVRP, where each route is limited by the amount g of *gas* (fuel) a vehicle can carry. Similar to LoadCVRP, the capacity g is equal for all vehicles.

▶ **Definition 2.7** (GasCVRP)**.** The *Gas-Capacitated Vehicle Routing Problem* (GasCVRP) is a GVRS with additional parameters $r \in \mathbb{N}$ and $g \in \mathbb{N}$. It asks whether there exists a routing (R, A) with $w(R) \leq r$, subject to $\Gamma = \Gamma_{\text{closed}} \wedge \Gamma_g$, where

$$\Gamma_g(R, A) \iff \forall p \in R : w(p) \leq g.$$ ⌟

Finally, LoadGasCVRP combines the constraints of LoadCVRP and GasCVRP.

▶ **Definition 2.8** (LoadGasCVRP)**.** The *Load-and-Gas-Capacitated Vehicle Routing Problem* (LoadGasCVRP) is a GVRS with additional parameters $r \in \mathbb{N}$, $\ell \in \mathbb{N}$, $\Lambda\colon C \to \mathbb{N}^+$, and $g \in \mathbb{N}$. It asks whether there exists a routing (R, A) with $w(R) \leq r$, subject to $\Gamma = \Gamma_{\text{closed}} \wedge \Gamma_{\ell,\Lambda} \wedge \Gamma_g$. ⌟

3 Uncapacitated Vehicle Routing

In this section, we establish the parameterized landscape around the Uncapacitated Vehicle Routing Problem. We show hardness of VRP when parameterized by the number of depots $|D|$ and vehicles k. In turn, we show that VRP is tractable when parameterized by the number of clients $|C|$. Finally, we present an algorithm which proves that EVRP (and thereby VRP) is tractable on graphs of bounded treewidth tw.

Note that for VRP routings, it suffices to find a set R of at most k closed walks that connect every client to a depot. This is because the predicate Γ does not pose any requirements on the assignment of vehicles A. In fact, an assignment A can be chosen based on R by arbitrarily selecting one of the vehicles which visit a given client. We thus do not mention the assignment A any further in this section.

Since MetricTSP is a special case of VRP where $C = V(G)$, $D = \{v\}$, and $k = 1$, we obtain the following result:

▶ **Theorem 3.1** ($\star$)**.** VRP *parameterized by* $|D| + k$ *is* paraNP*-hard.*

Note that the above observation still requires a large number of clients. In the next theorem, we see that we can efficiently solve VRP instances for constant numbers of clients.

▶ **Theorem 3.2** ($\star$)**.** *There is an algorithm that computes an optimal* VRP *routing in* $|C|^{\mathcal{O}(|C|)} \cdot n^{\mathcal{O}(1)}$ *steps.*

While the above algorithm outputs an optimal routing, it can also be used to solve the decision variant of VRP as defined in Definition 2.4. It also yields the following corollary:

▶ **Corollary 3.3** ($\star$)**.** *There is an algorithm that, given* $r \in \mathbb{N}$*, decides whether there exists a* VRP *routing of weight at most* r *in* $f(r) \cdot n^{\mathcal{O}(1)}$ *steps, for some computable function* f*.*

Next, we present an FPT algorithm for VRP parameterized by the treewidth. Our algorithm generalizes the prior work by Schierreich and Suchý [14] on the *Waypoint Routing Problem* (WRP). WRP can be formulated in terms of GVRS as follows:

▶ **Definition 3.4** (WRP)**.** The *Waypoint Routing Problem* (WRP) is a GVRS with $k = 1$, $D = \{s\}$, a designated vertex $t \in V(G)$, and additional parameters $r \in \mathbb{N}$ and $\kappa \colon E(G) \to \mathbb{N}$. It asks whether there exists a routing R (consisting of a single walk, due to $k = 1$) with $w(R) \le r$, subject to $\Gamma = \Gamma_t \wedge \Gamma_\kappa$, where

$$\Gamma_t(R, A) \iff \forall p \in R : p \text{ ends at } t,$$
$$\Gamma_\kappa(R, A) \iff \forall e \in E : e \text{ is used at most } \kappa(e) \text{ times in } R.$$ ⌟

▶ **Theorem 3.5** ([14])**.** *There is an algorithm that computes an optimal* WRP *routing in* $2^{\mathcal{O}(\text{tw})} \cdot n^{\mathcal{O}(1)}$ *steps, where* tw *denotes the treewidth of* G*.*

Schierreich and Suchý [14] obtain this result by reducing WRP to a normal form where $s = t$, effectively replacing the constraint Γ_t by Γ_{closed}. Next, they reformulate the problem to remove the parameter κ: Every edge $e \in E(G)$ is replaced by $\kappa(e)$ many parallel edges $e^1, e^2, \dots, e^{\kappa(e)}$, each with weight $w(e)$. Given the obtained multigraph G', WRP now asks to find a connected, Eulerian subgraph $H \subseteq G$, which contains the depot s and all clients C.

Since WRP is very similar to a single-vehicle case of VRP, we aim to adapt the techniques from [14] to our work. In particular, we slightly generalize the VRP problem and consider EVRP that incorporates edge capacities (see Definition 2.5), making WRP a proper special case of our problem. The introduction of edge capacities κ to VRP yields the same reformulation to multigraphs as found for WRP.

▶ **Lemma 3.6** (⋆). *Let G' be the multigraph obtained by replacing all edges e of G by $\kappa(e)$ parallel edges as described above. There is an* EVRP *routing R of weight $r \in \mathbb{N}$ in G if and only if there is an Eulerian subgraph $H \subseteq G'$ of weight $w(H) = r$ with $C \subseteq V(H)$ and at most k connected components, so that every client $c \in C$ is reachable from a depot $d \in D$.*

Based on Lemma 3.6, we reuse the term *routing* to refer to Eulerian subgraphs $H \subseteq G'$ that connect every client to a depot and contain at most k connected components. Furthermore, we shift the goal of our algorithm to finding a suitable subgraph $H \subseteq G'$. Before we can move on from G and κ, we need to make one more simplification to the problem: Although replicating all edges e by $\kappa(e)$ copies greatly increases the size of the input graph, this increase can be avoided by the following observation, which can be proven easily.

▶ **Lemma 3.7.** *Let $H \subseteq G'$ be an* EVRP *routing and $e \in E(H)$ be an edge with multiplicity greater than 2. Then $H - 2e$ is an* EVRP *routing and $w(H - 2e) \leq w(H)$.*

By repeatedly applying Lemma 3.7, we see that it suffices to solve EVRP instances with $\kappa(e) \leq 2$ for all edges e. In this case, the introduction of $\kappa(e)$ copies of an edge e can only increase the input size by a factor of 2. Therefore, we will not mention the simple graph G and the edge capacities κ anymore. Instead, we work on multigraphs where each edge can only be used once.

It seems like the algorithm from Theorem 3.5 cannot be used directly to solve EVRP. However, its general approach still applies to vehicle routing problems. The main difference is that in WRP, all partial solutions eventually merge to a connected subgraph containing the source vertex s, whereas in EVRP, there are multiple connected components, each of which contains one of the depots. It turns out that this observation directly translates to an FPT-algorithm for EVRP parameterized by treewidth and the number of connected components.

▶ **Theorem 3.8** (⋆). *There is an algorithm that computes an optimal* EVRP *routing in $\binom{|D|}{d^*}(\mathrm{tw} + d^*)^{\mathcal{O}(\mathrm{tw}+d^*)} \cdot n^{\mathcal{O}(1)}$ steps where $d^* = \min\{|D|, k\}$.*

Importantly, the optimization employed by [14] to improve the running time from $\mathrm{tw}^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ to $2^{\mathcal{O}(\mathrm{tw})} \cdot n^{\mathcal{O}(1)}$ cannot be used to improve Theorem 3.8. The optimization is based on the `reduce` operation from [3] that merges partial solutions which are equivalent when considering that they still have to be extended to the signature $\{\{s\}\}$ of a full solution. In Theorem 3.8, the signature at the root node is read from the partition $D[\emptyset]$, not $\{D\}$. Thus, a generalization of the setup by [3] to more complex partitions is needed to achieve a similar speedup.

While Theorem 3.8 yields that EVRP parameterized by $\mathrm{tw} + |D|$ is in FPT, it seems that the information tracked in the DP by [14] is almost sufficient for EVRP. This intuition is correct: By extending the weighted-partition setup of [3] and adapting the DP by [14], we manage to develop an algorithm for EVRP which is FPT when parameterized by just the treewidth of G. To adapt the setup we first introduce the coarsening relation and join operation on partitions. After presenting our extension of weighted partitions and adjusting the operations on weighted partitions by [3] to fit our definition, we can finally go on to present our tree decomposition based algorithm for EVRP. The complete list of definitions and adaptations can be found in the full version. We further show that the adapted operators correctly maintain and update each DP record, and using them, we derive the following theorem.

▶ **Theorem 3.9** (⋆). VRP *parameterized by* tw *is in* FPT.

4 Capacitated Vehicle Routing

In this section, we investigate the complexity of LoadCVRP, GasCVRP, and LoadGasCVRP. We observe that the three problems share most complexity characteristics. In particular, the requirement to adhere to vehicle capacities greatly increases the complexity. Whereas VRP admits an FPT algorithm when parameterized by the treewidth of G, all problems investigated in this section are NP-hard even on trees. We derive this hardness by encoding BinPacking instances in vehicle routing problems. In doing so, we also realize that the presence of zero-weight edges greatly impacts the tractability when parameterizing by the output weight r.

Next, we investigate the parameterized complexity with respect to the given vehicle capacities (ℓ for LoadCVRP and g for GasCVRP). We show paraNP-hardness by reduction from TrianglePacking for all three variants. This shows that capacitated vehicle routing problems do not only pose the challenge of "algebraic packing" as in BinPacking, but also one of "structural packing". We further provide paraNP-hardness results for LoadGasCVRP, when parameterized by treewidth, number of depots and either the load capacity or gas constraint, both by reduction from N3DMatching.

Finally, we find that the situation is not as dim when parameterizing by both the treewidth and the capacity value (for example $\text{tw} + \ell$ for LoadCVRP). While we cannot yet answer whether or not there can exist an FPT algorithm for these parameters, we provide an XP algorithm for LoadGasCVRP parameterized by $\text{tw} + \ell + g$, thereby ruling out paraNP-hardness. Additionally, our (to the best of our knowledge) novel FPT-algorithm for BinPacking parameterized by the capacity of each bin is a solid starting point for further analysis of the parameterized complexity of capacitated vehicle routing.

We start by extending some of our results from Section 3 to capacitated vehicle routing.

▶ **Theorem 4.1** ($\star$). *For each of LoadCVRP, GasCVRP, and LoadGasCVRP, there is an algorithm that computes an optimal routing in $|C|^{\mathcal{O}(|C|)} \cdot n^{\mathcal{O}(1)}$ steps.*

Note that the proof for Corollary 3.3 does not work for LoadCVRP in general as there is no sensible way to update the demand Λ when contracting zero-weight edges between two clients. This problem also occurs in LoadGasCVRP. Since client assignment is irrelevant for GasCVRP, and when no zero-weight edges are present the contraction step can be omitted, the proof of Corollary 3.3 applies analogously to the following statements.

▶ **Corollary 4.2.** *There is an algorithm that, given a GasCVRP instance and $r \in \mathbb{N}$, decides whether there exists a routing of weight at most r in $f(r) \cdot n^{\mathcal{O}(1)}$ steps, for some computable function f.*

▶ **Corollary 4.3.** *There is an algorithm that, given a LoadCVRP or LoadGasCVRP instance with no zero-weight edges and $r \in \mathbb{N}$, decides whether there exists a routing of weight at most r in $f(r) \cdot n^{\mathcal{O}(1)}$ steps, for some computable function f.*

Another insight is that limiting both k and the capacity constraint suffices for tractability, because the number of clients that can possibly be covered is limited.

▶ **Theorem 4.4** ($\star$). *All of the following problems are in FPT:*
- LoadCVRP *parameterized by $k + \ell$,*
- GasCVRP *parameterized by $k + g$, and*
- LoadGasCVRP *parameterized by $k + \ell$.*

4.1 Hardness from Bin Packing

Prior work regarding capacitated vehicle routing, such as by [12], has already observed the connection to BinPacking. Before we can formally capture the inclusion of BinPacking within vehicle routing, we first introduce the relevant prior work.

▶ **Definition 4.5** (BinPacking). A BinPacking instance consists of a finite set U of items, with sizes given by $s\colon U \to \mathbb{N}^+$, a bin capacity $B \in \mathbb{N}^+$, and a number of bins $k \in \mathbb{N}$. It asks whether there exists a partition $U_1, U_2, \dots, U_k$ of U such that for all $i \in [k]$ we have $\sum_{u \in U_i} s(u) \leq B$. Additionally, we define UnaryBinPacking as a BinPacking variant, in which all numerical inputs are encoded in unary. ⌟

Going forward, we assume that BinPacking instances do not contain items $u \in U$ with $s(u) > B$, as these instances can be rejected immediately.

▶ **Theorem 4.6** ([9]). *Both* BinPacking *and* UnaryBinPacking *are* NP*-complete.*

▶ **Theorem 4.7** ([10]). UnaryBinPacking *parameterized by the number of bins k is* W[1]*-hard.*

▶ **Theorem 4.8.** LoadCVRP*,* GasCVRP*, and* LoadGasCVRP *are* W[1]*-hard when parameterized by k, even with unit demands, unit weights, a single depot, and on trees.*

Proof. Similar to before, we give an extensive proof for LoadCVRP and briefly explain why it carries over to the other variants. Given a UnaryBinPacking instance (U, s, k, B), we construct a tree G with one central depot d and one branch for every item $u \in U$. To build such a branch, we add $s(u)$ vertices that form a path $v_u^1 v_u^2 \dots v_u^{s(u)}$ and connect this path to the depot via an edge $\{d, v_u^1\}$. The set of clients is the set of all vertices on these paths, that is, $C = \{v_u^i \mid u \in U \land i \in [s(u)]\}$. We then output the LoadCVRP instance

$$(G, w_1, D = \{d\}, C, k, \Lambda_1, \ell = B, r = 2|E(G)|),$$

where w_1 and Λ_1 are the unit weight and unit demand functions $w_1\colon e \mapsto 1$ and $\Lambda_1\colon c \mapsto 1$ respectively. Figure 1 shows an example of the reduction. Note that even though we are creating $s(u)$ vertices for every $u \in U$, this reduction still runs in polynomial time, as we are reducing from UnaryBinPacking.

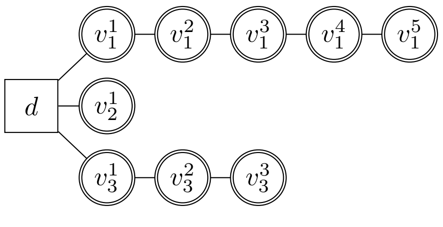

Figure 1 Example graph for $U = [3]$ with $s(1) = 5$, $s(2) = 1$, and $s(3) = 3$.

If there is a valid partition of U into k sets $U_1, \dots, U_k$, which serves as a solution to the BinPacking instance, for every $i \in [k]$ we let a vehicle cover all clients v_u^j for $u \in U_i$ and $j \in [s(u)]$. This can be achieved by having the vehicle traverse the tree from the depot d to $v_u^{s(u)}$ and back for each $u \in U_i$. This way, every edge is walked exactly twice and a vehicle $i \in [k]$ covers $\sum_{u \in U_i} s(u) \leq B$ clients.

Suppose in turn that there is a LoadCVRP routing $(R = \{p_1, p_2, \ldots, p_{k'}\}, A)$ for the given instance. Note that as G is a tree, $r = 2|E(G)|$ is just enough so that the vehicles can reach the vertices $\{v_u^{s(u)} \mid u \in U\}$. Thus, the path of an item $u \in U$ is traversed by exactly one vehicle. Given that $\ell = B$, the partition given by $U_i = \{u \in U \mid v_u^1 \in V(p_i)\}$ for all $i \in [k']$ abides the constraints of BinPacking.

The same reduction works without demands for $g = 2B$, as any vehicle can then again visit at most B clients. For LoadGasCVRP the reduction works with $\ell = B$ and $g = 2B$. ◀

The above translates to the following result regarding treewidth:

▶ **Corollary 4.9.** LoadCVRP*,* GasCVRP*, and* LoadGasCVRP *are* paraNP*-hard when parameterized by* $\text{tw} + |D|$ *and* W[1]*-hard when parameterized by* $\text{tw} + k + |D|$*, even with unit demands and unit weights, where* tw *denotes the treewidth of* G*.*

When mapping BinPacking instances to vehicle routing, the number of bins corresponds to the number of vehicles, and the bin capacity corresponds to the capacity of each vehicle. As we see later in Corollary 4.20, BinPacking parameterized by B is in FPT. Thus, an FPT-reduction from BinPacking mapping a value from B to some vehicle capacity does not yield any hardness result. CVRP instances parameterized by the vehicle capacities could potentially be reduced to BinPacking parameterized by B, but such a reduction is unlikely to be applicable to the general case, as vehicle routing problems not only require a numerical partitioning as with BinPacking, but also a structural one found in TrianglePacking. Still, we return to this intuition in Section 4.4, where we develop an XP algorithm for LoadGasCVRP.

4.2 Hardness from Triangle Packing

In this section, we show that TrianglePacking is contained within the three CVRP variants we consider.

▶ **Definition 4.10** (TrianglePacking)**.** A TrianglePacking instance consists of an undirected graph G such that $|V(G)| = 3q$ vertices, for some integer $q \in \mathbb{N}$. It asks whether there exists a partition $V_1, V_2, \ldots, V_q$ of $V(G)$ so that for each $i \in [q]$, $G[V_i] \simeq K_3$. ⌟

▶ **Theorem 4.11** ([9])**.** TrianglePacking *is* NP*-complete.*

Observe that a partition into triangle graphs is equivalent to a fleet of vehicles that each drive a small tour of just three vertices. We use this intuition in the following reduction.

▶ **Theorem 4.12** ($\star$)**.** *All of the following problems are* paraNP*-hard, even with unit demands and weights:*

- LoadCVRP *parameterized by* ℓ*,*
- GasCVRP *parameterized by* g*, and*
- LoadGasCVRP *parameterized by* $\ell + g$*.*

4.3 Hardness from Numerical 3D Matching

In this section we use a reduction from N3DMatching to show that LoadGasCVRP is strongly NP-hard even on trees with constant number of depots, load capacity and unit demands. The reduction can also be adapted to show hardness on trees with constant number of depots, a constant gas constraint and unit edge weights.

▶ **Definition 4.13** (N3DMatching). A *Numerical 3-Dimensional Matching* instance consists of three disjoint sets $X, Y, Z \subseteq \mathbb{N}$, each containing m elements, and a bound $b \in \mathbb{N}$ with $\sum_{a \in X \cup Y \cup Z} a = m \cdot b$. It asks wether $X \cup Y \cup Z$ can be partitioned into m disjoints sets $A_1, \ldots, A_m$, such that each A_i contains exactly one element from each of X, Y, Z and $\sum_{a \in A_i} a = b$. ⌟

▶ **Theorem 4.14** ([9]). N3DMatching *is strongly* NP*-complete.*

▶ **Theorem 4.15** ($\star$). LoadGasCVRP *is strongly* NP*-hard, even on stars with one depot, constant load capacities and unit demands.*

To show the hardness of LoadGasCVRP on stars with one depot and a constant gas constraint, the construction used in the proof of Theorem 4.15 can be modified. We use client demands instead of edge weights to encode the numbers to be matched and their type in a given N3DMatching instance, changing the load and gas constraints accordingly.

▶ **Corollary 4.16.** LoadGasCVRP *is strongly* NP*-hard, even on stars with one depot and a constant gas constraint.*

4.4 Tractability for Constant Treewidth and Vehicle Capacities

In this section we investigate LoadGasCVRP parameterized by $\mathrm{tw} + \ell + g$. Note that the observation from Lemma 3.7 does not hold once vehicles need to abide capacities, as witnessed in Figure 2.

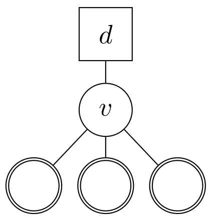

Figure 2 A LoadCVRP instance with $\ell = 1$ where each routing has to use the edge $\{d, v\}$ more than twice.

Thus, the dynamic programming devised to prove Theorem 3.9 cannot directly be applied to LoadGasCVRP. Instead, we track the number of walks with a given set of characteristics separately, leading to an XP running time. Note however, that a similar result to Lemma 3.7 holds for each vehicle individually:

▶ **Lemma 4.17** ($\star$). *Let (R, A) be a* LoadGasCVRP *routing, let $p \in R$ and let $H \subseteq G$ be the undirected multigraph representing p. If there is an edge $e \in E(H)$ with multiplicity greater than two, then the walk p' obtained from $H - 2e$ can be used instead of p in R to form a* LoadGasCVRP *routing R' with $w(R') \leq w(R)$.*

Before we formally define the state of the dynamic programming algorithm on the tree decomposition of a given instance for this section, we divert back to BinPacking. We define and solve an extended version of BinPacking, which abstracts from the combination of different sets of walks in order to form new walks fulfilling some desired characteristics. In the algorithm, this operation will be needed for the computation at the Join Nodes of the tree decomposition.

In particular, we encode each walk as a BinPacking item with a *multidimensional size* given by the number of clients and distance traveled. Accordingly, the bins have multidimensional capacities matching the characteristics of the desired walks. As the walks of a partial solution might have different characteristics, there are several different *bin kinds*. In addition, each walk has a *fingerprint* given by the endpoints of the walk and whether or not it contains a depot. Each bin kind expects a combination of fingerprints fulfilling a certain predicate, according to the endpoints of the desired walk.

▶ **Definition 4.18** (HetMdFpBinPacking)**.** A *Heterogeneous Multidimensional Fingerprint Bin Packing* instance consists of

- a finite set U of items,
- a finite set $\mathcal{F}$ of fingerprints,
- a dimension $d \in \mathbb{N}^+$,
- a number of bin kinds $m \in \mathbb{N}^+$,
- bin capacities $B_1, B_2, \dots, B_m \in \mathbb{N}^d$ (with $\|B_1\|_1 \geq \|B_2\|_1 \geq \dots \geq \|B_m\|_1$),
- available bin counts $k_1, k_2, \dots, k_m$,
- predicates deciding the validity of a multiset of fingerprints $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m : \mathbb{N}^{\mathcal{F}} \to \{0,1\}$,
- item sizes $s\colon U \to \mathbb{N}^d \setminus \{0\}$ with $\|s(u)\|_1 \leq \|B_1\|_1$ for all $u \in U$, and
- item fingerprints $F\colon U \to \mathcal{F}$.

It asks whether there exists a partition $\{U_{i,j}\}_{i\in[m],j\in[k_i]}$ of U such that for all i, j

- the bin capacity is not exceeded: $\sum_{u\in U_{i,j}} s(u) \leq B_i$, and
- the multiset of fingerprints is valid: $\mathcal{V}_i(\llbracket F(u) \mid u \in U_{i,j} \rrbracket) = 1$. ⌟

▶ **Theorem 4.19** (⋆)**.** *There is an algorithm that, given a* HetMdFpBinPacking *instance*

$$\mathcal{B} = (U, \mathcal{F}, d, m, (B_1, B_2, \dots, B_m), (k_1, k_2, \dots, k_m), (\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m), s, F),$$

computes a feasible partition or concludes that there is none in $f(|\mathcal{F}| + d + m + \|B_1\|_1) \cdot (T(|U|) + |\mathcal{B}|^{\mathcal{O}(1)})$ *steps, where* $T\colon \mathbb{N} \to \mathbb{N}$ *is an upper bound on the time complexity of the predicates* $\{\mathcal{V}_i\}_{i\in[m]}$*, for some computable function* f*.*

Note that removing all the additions of HetMdFpBinPacking, the above algorithm can also be used for regular BinPacking.

▶ **Corollary 4.20.** BinPacking *parameterized by the bin capacity* B *is in* FPT*.*

With all preliminaries out of the way, we present the dynamic programming algorithm for LoadGasCVRP. Given a LoadGasCVRP instance $(G, w, D, C, k, \ell, \Lambda, g, r)$ and a nice tree decomposition $\mathcal{T}$ of G, we track partial solutions of the following form:

▶ **Definition 4.21.** Let $t \in \mathcal{T}$. We call a walk p in $G_t^{\downarrow}$ *detached* if $V(p) \cap X_t = \emptyset$ and $V(p) \cap D \neq \emptyset$. We call p *attached* when both endpoints of p are in X_t (but we don't require anything with regards to D). For $u, v \in V(G), \lambda, w \in \mathbb{N}$, an attached u-v-walk with $\Lambda(p) = \lambda$ and $w(p) = w$ is called a (u, v, λ, w)-walk.

Let $S_t = \{s\colon X_t^2 \times [\ell]_0 \times [g]_0 \to \mathbb{N}\}$ be the set of all multisets of combinations of walk endpoints, clients covered, and weight. We refer to the elements $s \in S_t$ as *counters* and abbreviate $\Lambda(s) = \sum_{u,v,\lambda,w} \lambda \cdot s(u, v, \lambda, w)$ and $w(s) = \sum_{u,v,\lambda,w} w \cdot s(u, v, \lambda, w)$.

Let $X \subseteq X_t \cap C$, $c \leq k$, and $s, s^* \in S_t$. We call (X, c, s, s^*) a *solution signature* at t. A set of walks $R = \{p_1, p_2, \dots, p_{k'}\}$ in $G_t^{\downarrow}$ and assignment of clients $A\colon C \cap (X \cup (V_t^{\downarrow} \setminus X_t)) \to [k']$ is a *partial solution* compatible with (X, c, s, s^*) at t, if

- for all clients $c \in C \cap (X \cup (V_t^{\downarrow} \setminus X_t))$, $c \in V(p_{A(c)})$,
- all walks $p \in R$ are either detached or attached,
- there are exactly c detached walks in R,
- for each $u, v \in X_t, \lambda \in [\ell]_0, w \in [g]_0$,
  - there are exactly $s(u, v, \lambda, w)$ many (u, v, λ, w)-walks $p \in R$ with $V(p) \cap D = \emptyset$, and
  - there are exactly $s^*(u, v, \lambda, w)$ many (u, v, λ, w)-walks $p \in R$ with $V(p) \cap D \neq \emptyset$. ⌟

We use dynamic programming to compute the minimum weight $dp[t, X, c, s, s^*] \in \mathbb{N} \cup \{\infty\}$ of all partial solutions R compatible with (X, c, s, s^*) at all $t \in \mathcal{T}$ and for all solution signatures (X, c, s, s^*) at t. The observant reader might have noticed that since the counters $s \in S_t$ map to the natural numbers, the set S_t is infinite when $X_t \neq \emptyset$. However, we know from Lemma 4.17 that it suffices to look for routings where each of the k walks contain each edge of G at most twice. As there are only finitely many such routings, only finitely many partial solutions leading to these routings can be witnessed at the bags of $\mathcal{T}$. Thus, there are only finitely many counters which are relevant for finding an optimal routing.

First, if the number of covered clients indicated by a counter s is greater than the number of available clients, we know that no compatible partial solution exists. More formally, if

$$\Lambda(s + s^*) > |C \cap (X \cup (V_t^{\downarrow} \setminus X_t))|,$$

then $dp[t, X, \cdot, s, s^*] = \infty$. Note that the above values might still not be equal, as some of the clients in $V_t^{\downarrow}$ might be covered by detached walks. This means that we can limit the computation of dp to only those counters $s \in S_t$ with $s(\cdot, \cdot, \lambda, \cdot) \leq |C|$ for all $\lambda \geq 1$.

This still leaves the value of s unbounded for $\lambda = 0$. However, the dp values for counters with $s(u, v, 0, w) > 1$ can be reduced to the case where $s(u, v, 0, w) = 1$. The fact that a partial solution (R, A) containing a $(u, v, 0, w)$-walk p exists suffices to find the minimum weight partial solution for $s(u, v, 0, w) > 1$. Such a solution can be obtained by repeatedly inserting copies of p into R, which increases the weight of R by $(s(u, v, 0, w) - 1) \cdot w$. Thus, we only need to compute $dp[t, X, c, s, s^*]$ when $s(\cdot, \cdot, 0, \cdot), s^*(\cdot, \cdot, 0, \cdot) \leq 1$.

Before we can start to describe the computation of dp, we need to introduce the concept of *counter reducibility.*

▶ **Definition 4.22.** Let $t \in T$ and $s_1, s_1^*, s_2, s_2^* \in S_t$. The pair of counters (s_1, s_1^*) is reducible to (s_2, s_2^*), denoted as $(s_1, s_1^*) \rightsquigarrow (s_2, s_2^*)$, if for every $X \subseteq X_t$, $c \leq k$, and partial solution (R_1, A_1) compatible with (X, c, s_1, s_1^*) at t, there is a partial solution (R_2, A_2) compatible with (X, c, s_2, s_2^*) at t which can obtained by merging walks of R_1 and updating A_1 accordingly. ⌟

▶ **Lemma 4.23** ($\star$). *Counter reducibility can be decided in $f(|X_t| + \ell + g + z) \cdot n^{\mathcal{O}(1)}$ steps for some function f, where $z = \sum_{u,v \in X_t}(s_2(u, v, 0, 0) + s_2^*(u, v, 0, 0))$.*

The computation of dp then depends on the node type of t:

Leaf Node

If t is a leaf node, then $X_t = \emptyset$, so X, s and s^* must all be empty too. As $V_t^{\downarrow} = \emptyset$, there cannot be any detached walks, so

$$dp[t, X, c, s, s^*] = \begin{cases} 0, & \text{if } c = 0, \\ \infty, & \text{otherwise.} \end{cases}$$

Introduce Vertex Node

Suppose that t is an introduce vertex node with child node t' such that $X_t = X_{t'} \cup \{v\}$. As v is an isolated vertex in $G_t^{\downarrow}$, any attached walk p that contains v must be the empty walk. Thus, for all $u \neq v$ and $w > 0$ we must have $s(u, v, \cdot, \cdot) = s(v, u, \cdot, \cdot) = s^*(u, v, \cdot, \cdot) = s^*(v, u, \cdot, \cdot) = s(v, v, \cdot, w) = s^*(v, v, \cdot, w) = 0$. Depending on whether or not $v \in D$, the empty walks at v are counted in s or s^* respectively. Suppose that $v \notin D$, then $s^*(v, v, \cdot, \cdot) = 0$. One of the empty walks at v should cover a client if and only if v is a client and contained in X. More formally, for all $\lambda \geq 1$ we have $s(v, v, \lambda, 0) = \mathbb{1}(\lambda = \Lambda(v) \wedge v \in X)$. Lastly, the value $s(v, v, 0, 0)$ can take any value. If all of the above constraints are satisfied, we set $dp[t, X, c, s, s^*] = dp[t', X \setminus \{v\}, c, s_{\downarrow X_{t'}}, s^*_{\downarrow X_{t'}}]$. Otherwise there is no compatible partial solution, and the dp entry is set to ∞. The case $v \in D$ follows analogously with the requirements for s and s^* swapped.

Introduce Edge Node

Suppose that t is an introduce edge node labeled with the edge $e = \{u, v\}$ with a child t'. As seen in Lemma 4.17, any walk in a minimum weight partial solution compatible with (X, c, s, s^*) at t traverses e at most twice. Thus, when removing e, a walk p gets split into at most three subwalks. Let $s', s'^* \in S_{t'}$ and $c_e \in \mathbb{N}$ such that $\max\{c_e, |s'| + |s'^*|\} \leq 3 \cdot (|s| + |s^*|)$. Then a partial solution compatible with (X, c, s, s^*) at t can be constructed from a partial solution compatible with (X, c, s', s'^*) at t' by inserting c_e copies of e if $(s' + c_e \cdot [\![(u, v, 0, w(u, v))]\!], s'^*) \rightsquigarrow (s, s^*)$. Thus, we set

$$dp[t, X, c, s, s^*] = \min_{s', s'^*, c_e} dp[t', X, c, s', s'^*] + c_e \cdot w(u, v),$$

where s', s'^*, c_e range over the values described above.

Forget Node

Suppose that t is a forget node with child node t' such that $X_t = X_{t'} \setminus \{v\}$. A partial solution compatible with (X, c, s, s^*) can interact with v in several ways.

First, there can be detached walks containing v, which are included in the count c. For a given count $c' \leq c$ of detached walks that are also detached from t', let $s_v \in S_{t'}$ be a counter which has $s_v(a, b, \cdot, \cdot) = 0$ whenever $a \neq v$ or $b \neq v$ and $\sum s_v(v, v, \cdot, \cdot) = c - c'$. There are at most $(c - c' + 1)^{(\ell+1)\cdot(g+1)}$ such counters s_v. As the detached walks need to contain a depot, we consider the counter s_v as an addition to s^*.

Second, the walks counted in s and s^* can contain v. However, any such walk cannot start or end at v, as it would otherwise not be attached to X_t. Thus, the counters for t' remain unchanged by this interaction.

In both cases, if $v \in C$, it must be covered regardless of X. Therefore, we set

$$dp[t, X, c, s, s^*] = \min_{c', s_v} dp[t', X \cup (C \cap \{v\}), c', s, s^* + s_v]$$

where c and s_v range over the counts and counters described above.

Join Node

Suppose that t is a join node with children t_1 and t_2. A partial solution compatible with (X, c, s, s^*) is composed of partial solutions from both t_1 and t_2 to cover the clients in $V^\downarrow_{t_1}$ and $V^\downarrow_{t_2}$ respectively. A walk from a partial solution at t might switch between $V^\downarrow_{t_1}$ and $V^\downarrow_{t_2}$ a number of times, so it is split into multiple walks at the children. Suppose that a walk p in $G^\downarrow_t$ is split into the walks $p_1, p_2, \ldots, p_q$. First, the clients covered by p are distributed among the p_i. There are at most ℓ such client-connecting walks. All walks p_i that do not cover a client are only needed to connect the start and end of two client-connecting walks. As there are $|X_t|$ vertices that could be reached, at most $|X_t|$ walks are needed in between two client-connecting walks. In total, there are thus at most ℓ client-connecting walks and $(\ell + 1) \cdot |X_t|$ walks connecting the endpoints of those. Therefore, we limit the counters

$s_1, s_1^* \in S_{t_1}, s_2, s_2^* \in S_{t_2}$ to those which are bounded above by $(\ell+1) \cdot |X_t| \cdot (|s| + |s^*|)$. Similarly to before, only the counters which satisfy $(s_1 + s_2, s_1^* + s_2^*) \rightsquigarrow (s, s^*)$ are considered. Note that the clients in X are split between the two partial solutions so that the partial solution in t_1 covers some subset $X_1 \subseteq X$ and leaves $X \setminus X_1$ to t_2's solution. Similarly, the number of detached walks is split into $c_1 \leq c$ and $c - c_1$. Bringing it all together, we set

$$dp[t, X, c, s, s^*] = \min_{\substack{X_1 \subseteq X, c_1 \leq c,\\ s_1, s_1^*, s_2, s_2^*}} dp[t_1, X_1, c_1, s_1, s_1^*] + dp[t_2, X \setminus X_1, c - c_1, s_2, s_2^*],$$

where s_1, s_1^*, s_2, s_2^* range over the values described above.

▶ **Theorem 4.24** ($\star$). LoadGasCVRP *parameterized by* $\mathrm{tw} + \ell + g$ *is in* XP*.*

Furthermore, the algorithm from Theorem 4.24 can be modified to yield similar results for LoadCVRP and GasCVRP. In particular, the counters S_t for each node $t \in \mathcal{T}$ should only track the endpoints of each walk and one of the two constraints.

▶ **Corollary 4.25.** LoadCVRP *parameterized by* $\mathrm{tw} + \ell$ *and* GasCVRP *parameterized by* $\mathrm{tw} + g$ *are in* XP*.*

5 Conclusion

We have presented detailed tractability results for vehicle routing problems regarding the parameters given in the input and the treewidth of the underlying network. We have shown complete results for all mentioned parameters in the uncapacitated case. For VRP treewidth as a parameter suffices for the existence of FPT algorithms. On the other hand, once any form of capacity constraints is introduced, vehicle routing becomes hard even on trees. Combining treewidth with other parameters is only sensible for parameters which are not sufficient for FPT algorithms on their own, namely the capacity, k and $|D|$. Out of these parameters, only the inclusion of all present capacity parameters yields tractability.

While we also proved complexity bounds of the capacitated vehicle routing variants, some questions remain unanswered. We have provided an XP algorithm for treewidth and capacity as a combined parameter. However, the existence of an FPT algorithm for the problem could neither be ruled out nor proven in this work. Additionally, while the W[1]-hardness for the parameter $\mathrm{tw} + k + |D|$ is likely to rule out an FPT-algorithm, future work could explore XP algorithms for this parameter.

References

1 Saeed Akhoondian Amiri, Klaus-Tycho Foerster, and Stefan Schmid. Walking through waypoints. *Algorithmica*, 82(7):1784–1812, 2020. `doi:10.1007/S00453-020-00672-Z`.

2 Matthias Bentert, Fedor V. Fomin, Petr A. Golovach, Tuukka Korhonen, William Lochet, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Kirill Simonov. Packing short cycles. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 1425–1463. SIAM, 2025. `doi:10.1137/1.9781611978322.45`.

3 Hans L Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. `doi:10.1016/J.IC.2014.12.008`.

4 Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & industrial engineering*, 99:300–313, 2016. `doi:10.1016/J.CIE.2015.12.007`.

5 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. doi:10.1007/978-3-319-21275-3.

6 G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, October 1959.

7 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, and Sudeshna Kolay. An eth-tight exact algorithm for euclidean tsp. *SIAM Journal on Computing*, 52(3):740–760, 2023. doi:10.1137/22M1469122.

8 Michelle Döring, Jan Fehse, Tobias Friedrich, Paula Marten, Niklas Mohrin, Kirill Simonov, Farehe Soheil, Jakob Timm, and Shaily Verma. Parameterized complexity of vehicle routing. *arXiv preprint arXiv:2509.10361*, 2025.

9 Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

10 Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013. doi:10.1016/j.jcss.2012.04.004.

11 Andrea Mor and Maria Grazia Speranza. Vehicle routing problems over time: a survey. *Annals of Operations Research*, 314(1):255–275, 2022. doi:10.1007/S10479-021-04488-0.

12 Ted K Ralphs, Leonid Kopman, William R Pulleyblank, and Leslie E Trotter. On the capacitated vehicle routing problem. *Mathematical programming*, 94:343–359, 2003. doi:10.1007/S10107-002-0323-0.

13 Neil Robertson and Paul D Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

14 Šimon Schierreich and Ondřej Suchý. Waypoint routing on bounded treewidth graphs. *Information Processing Letters*, 173:106165, 2022. doi:10.1016/j.ipl.2021.106165.