Parameterized complexity of finding a spanning tree with minimum reload cost diameter

Julien Baste, Didem Gözüpek, Christophe Paul, Ignasi Sau, Mordechai Shalom, Dimitrios M. Thilikos

Parameterized complexity of finding a spanning tree with minimum reload cost diameter*

Julien Baste[†] Didem Gözüpek[‡] Christophe Paul[§] Ignasi Sau[§]
Mordechai Shalom[¶][‖] Dimitrios M. Thilikos[§][**]

Abstract

We study the minimum diameter spanning tree problem under the reload cost model (Diameter-Tree for short) introduced by Wirth and Steffan (2001). In this problem, given an undirected edge-colored graph G, reload costs on a path arise at a node where the path uses consecutive edges of different colors. The objective is to find a spanning tree of G of minimum diameter with respect to the reload costs. We initiate a systematic study of the parameterized complexity of the Diameter-Tree problem by considering the following parameters: the cost of a solution, and the treewidth and the maximum degree Δ of the input graph. We prove that Diameter-Tree is para-NP-hard for any combination of two of these three parameters, and that it is FPT parameterized by the three of them. We also prove that the problem can be solved in polynomial time on cactus graphs. This result is somehow surprising since we prove Diameter-Tree to be NP-hard on graphs of treewidth two, which is best possible as the problem can be trivially solved on forests. When the reload costs satisfy the triangle inequality, Wirth and Steffan (2001) proved that the problem can be solved in polynomial time on graphs with $\Delta = 3$, and Galbiati (2008) proved that it is NP-hard if $\Delta = 4$. Our results show, in particular, that without the requirement of the triangle inequality, the problem is NP-hard if $\Delta = 3$, which is also best possible. Finally, in the case where the reload costs are polynomially bounded by the size of the input graph, we prove that Diameter-Tree is in XP and W[1]-hard parameterized by the treewidth plus Δ.

Keywords: reload cost problems; minimum diameter spanning tree; parameterized complexity; FPT algorithm; treewidth; dynamic programming.

*Emails of authors: `julien.baste@uni-ulm.de`, `didem.gozupek@gtu.edu.tr`, `paul@lirmm.fr`, `sau@lirmm.fr`, `cmshalom@telhai.ac.il`, `sedthilk@thilikos.info`.

Work supported by the research program CNRS/TUBITAK grant no.114E731, TUBITAK 2221 programme, and by projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010).

An extended abstract of this work appeared in the *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (**IPEC**), volume 89 of LIPIcs, pages 3:1–3:12, Vienna, Austria, September **2017**.*

[†]Université de Montpellier, LIRMM, Montpellier, France.

[‡]Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey.

[§]CNRS, LIRMM, Université de Montpellier, France.

[¶]TelHai College, Upper Galilee, 12210, Israel.

[‖]Department of Industrial Engineering, Boğaziçi University, Istanbul, Turkey.

[**]Department of Mathematics, National and Kapodistrian University of Athens, Athens, Greece.

1 Introduction

Numerous network optimization problems can be modeled by edge-colored graphs. Wirth and Steffan introduced in [33] the concept of *reload cost*, which refers to the cost that arises in an edge-colored graph while traversing a vertex via two consecutive edges of different colors. The value of the reload cost depends on the colors of the traversed edges. Although the reload cost concept has many important applications in telecommunication networks, transportation networks, and energy distribution networks, it has surprisingly received attention only recently.

In heterogeneous communication networks, routing requires switching among different technologies such as cables, fibers, and satellite links. Due to data conversion between incompatible subnetworks, this switching causes high costs, largely outweighing the cost of routing the packets within each subnetwork. The recently popular concept of vertical handover [11], which allows a mobile user to have undisrupted connection during transitioning between different technologies such as 3G (third generation) and wireless local area network (WLAN), constitutes another application area of the reload cost concept. Even within the same technology, switching between different service providers incurs switching costs. Another paradigm that has received significant attention in the wireless networks research community is *cognitive radio networks* (CRN), a.k.a. *dynamic spectrum access networks*. Unlike traditional wireless technologies, CRNs operate across a wide frequency range in the spectrum and frequently require frequency switching; therefore, the frequency switching cost is indispensable and of paramount importance. Many works in the CRNs literature focused on this frequency switching cost from an application point of view (for instance, see [1, 3–5, 14, 21, 31]) by analyzing its various aspects such as delay and energy consumption. Operating in a wide range of frequencies is indeed a property of not only CRNs but also other 5G technologies. Hence, applications of the reload cost concept in communication networks continuously increase. In particular, the energy consumption aspect of this switching cost is especially important in the recently active research area of green networks, which aim to tackle the increasing energy consumption of information and communication technologies [6, 8].

The concept of reload cost also finds applications in other networks such as transportation networks and energy distribution networks. For instance, a cargo transportation network uses different means of transportation. The loading and unloading of cargo at junction points is costly and this cost may even outweigh the cost of carrying the cargo from one point to another [15]. In energy distribution networks, reload costs can model the energy losses that occur at the interfaces while transferring energy from one type of carrier to another [15].

Recent work in the literature focused on numerous problems related to the reload cost concept: the minimum reload cost cycle cover problem [17], the problems of finding a path, trail or walk with minimum total reload cost between two given vertices [20], the problem of finding a spanning tree that minimizes the sum of reload costs of all paths between all pairs of vertices [18], various path, tour, and flow problems related to reload costs [2], the minimum changeover cost arborescence problem [16, 22, 23, 25], and problems related to finding a proper edge coloring of the graph so that the total reload cost is minimized [24].

The work in [33], which introduced the concept of reload cost, focused on the following problem, called Minimum Reload Cost Diameter Spanning Tree (Diameter-Tree for short), and which is the one we study in this paper: given an undirected graph $G = (V, E)$ with a (non necessarily proper) edge-coloring $\chi : E(G) \to X$ and a reload cost function $c : X^2 \to \mathbb{N}_0$, find a spanning tree of G with minimum diameter with respect to the reload costs (see Section 2 for the formal definitions).

This problem has important applications in communication networks, since forming a spanning tree is crucial for broadcasting control traffic such as route update messages. For instance, in a multi-hop cognitive radio network where a frequency is assigned to each wireless link depending on availability of spectrum bands, delay-aware broadcasting of control traffic necessitates the forming of a spanning tree by taking the delay arising from frequency switching at every node into account. Cognitive nodes send various control information messages to each other over this spanning tree. A spanning tree with minimum reload cost diameter in this setting corresponds to a spanning tree in which the maximum frequency switching delay between any two nodes on the tree is minimized. Since control information is crucial and needs to be sent to all other nodes in a timely manner, ensuring that the maximum delay is minimum is vital in a cognitive radio network.

Wirth and Steffan [33] proved that Diameter-Tree is inapproximable within a factor better than 3 (in particular, it is NP-hard), even on graphs with maximum degree 5. They also provided a polynomial-time exact algorithm for the special case where the maximum degree is 3 and the reload costs satisfy the triangle inequality. Galbiati [15] showed stronger hardness results for this problem, by proving that even on graphs with maximum degree 4, the problem cannot be approximated within a factor better than 2 if the reload costs do not satisfy the triangle inequality, and cannot be approximated within any factor better than 5/3 if the reload costs satisfy the triangle inequality. The complexity of Diameter-Tree (in the general case) on graphs with maximum degree 3 was left open.

Our results. In this article we initiate a systematic study of the complexity of the Diameter-Tree problem, with special emphasis on its parameterized complexity for several choices of the parameters. Namely, we consider any combination of the parameters k (the cost of a solution), tw (the treewidth of the input graph), and Δ (the maximum degree of the input graph). We would like to note that these parameters have practical importance in communication networks. Indeed, besides the natural parameter k, whose relevance is clear, many networks that model real-life situations appear to have small treewidth [27, 30]. On the other hand, the degree of a node in a network is related to its number of transceivers, which are costly devices in many different types of networks such as optical networks [29]. For this reason, in practice the maximum degree of a network usually takes small values.

Before elaborating on our results, a summary of them can be found in Table 1.

Problem	Parameterized complexity with parameter				Polynomial cases
	$k + \mathsf{tw}$	$k + \Delta$	$\mathsf{tw} + \Delta$	$k + \mathsf{tw} + \Delta$	
Diameter-Tree	NPh for $k = 9, \mathsf{tw} = 2$ (Thm 1)	NPh for $k = 0, \Delta = 3$ (Thm 2)	NPh for $\mathsf{tw} = 3, \Delta = 3$ (Thm 3)	FPT (Thm 5)	in P on cacti (Thm ??)
Diameter-Tree with poly costs	✓	✓	XP (Thm 5) W[1]-hard (Thm 6)	✓	✓

Table 1: Summary of our results, where k, tw, Δ denote the cost of the solution, the treewidth, and the maximum degree of the input graph, respectively. NPh stands for NP-hard. The symbol '✓' denotes that the result above still holds for polynomial costs.

We first prove, by a reduction from 3-Sat, that Diameter-Tree is NP-hard on outerplanar graphs (which have treewidth at most 2) with only one vertex of degree greater than 3, even with three different costs that satisfy the triangle inequality, and $k = 9$. Note that, in the case where the costs satisfy the triangle inequality, having only one vertex of degree greater than 3 is best possible, as if all vertices have degree at most 3, the problem can be solved in polynomial time [33]. Note also that the bound on the treewidth is best possible as well, since the problem is trivially solvable on graphs of treewidth 1, i.e., on forests.

Toward investigating the border of tractability of the problem with respect to treewidth, we exhibit a polynomial-time algorithm on a relevant subclass of the graphs of treewidth at most 2: cactus graphs. This algorithm is quite involved and, in a nutshell, processes in a bottom-up manner the *block tree* of the given cactus graph, and uses at each step of the processing an algorithm that solves 2-Sat as a subroutine.

Back to hardness results, we also prove, by a reduction from a restricted version of 3-Sat, that Diameter-Tree is NP-hard on graphs with $\Delta \leqslant 3$, even with only two different costs, $k = 0$, and a bounded number of colors. In particular, this settles the complexity of the problem on graphs with $\Delta \leqslant 3$ in the general case where the triangle inequality is not necessarily satisfied, which had been left open in previous work [15, 33]. Note that $\Delta \leqslant 3$ is best possible, as Diameter-Tree can be easily solved on graphs with $\Delta \leqslant 2$.

As our last NP-hardness reduction, we prove, by a reduction from Partition, that the Diameter-Tree problem is NP-hard on planar graphs with $\mathsf{tw} \leqslant 3$ and $\Delta \leqslant 3$.

The above hardness results imply that the Diameter-Tree problem is para-NP-hard for any combination of two of the three parameters k, tw, and Δ. On the positive side, we show that Diameter-Tree is FPT parameterized by the three of them, by using a (highly nontrivial) dynamic programming algorithm on a tree decomposition of the input graph.

Since our para-NP-hardness reduction with parameter $\mathsf{tw} + \Delta$ is from Partition, which is a typical example of a *weakly* NP-complete problem [19], a natural question is whether Diameter-Tree, with parameter $\mathsf{tw} + \Delta$, is para-NP-hard, XP, W[1]-hard, or

FPT when the reload costs are *polynomially bounded* by the size of the input graph. We manage to answer this question completely: we show that in this case the problem is in XP (hence *not* para-NP-hard) and W[1]-hard parameterized by $\mathsf{tw} + \Delta$. The W[1]-hardness reduction is from the UNARY BIN PACKING problem parameterized by the number of bins, proved to be W[1]-hard by Jansen *et al.* [26].

Altogether, our results provide an accurate picture of the (parameterized) complexity of the DIAMETER-TREE problem.

Organization of the paper. We start in Section 2 with some brief preliminaries about graphs, the DIAMETER-TREE problem, parameterized complexity, and tree decompositions. In Section 3 we provide the para-NP-hardness results. In Section 4 we present the polynomial-time algorithm on cactus graphs, and in Section 5 we present the FPT algorithm on general graphs parameterized by $k + \mathsf{tw} + \Delta$. In Section 6 we focus on the case where the reload costs are polynomially bounded. Finally, we conclude the article in Section 7.

2 Preliminaries

Graphs and sets. We use standard graph-theoretic notation, and we refer the reader to [12] for any undefined term. Given a graph G and a set $S \subseteq V(G)$, we define $\mathsf{adj}_G(S)$ to be the set of edges of G that intersect S, i.e., those edges that have at least one endpoint in S. We also define $N_G[S] = S \cup \{x \mid \exists y \in S : \{x, y\} \in E(G)\}$. Given a graph G and a vertex $v \in V(G)$ with exactly two neighbors u and w, *dissolving* v is the operation that consists in removing v, $\{u, v\}$, and $\{v, w\}$ and adding the edge $\{u, w\}$. For a graph G and an edge $e \in E(G)$, we denote $G - e = (V(G), E(G)\backslash\{e\})$. Given two integers i and j with $i \leqslant j$, we use $[i, j]$ to denote the set of integers k such that $i \leqslant k \leqslant j$. We use the shorthand notation $[i]$ for $[1, i]$.

Reload costs and definition of the problem. For reload costs, we follow the notation and terminology defined by [33]. We consider edge-colored graphs $G = (V, E)$, where the colors are taken from a finite set X and the coloring function is $\chi : E(G) \to X$. The reload costs are given by a nonnegative function $c : X^2 \to \mathbb{N}_0$, which we assume to be symmetric. The cost of traversing two incident edges e_1, e_2 is $c(e_1, e_2) := c(\chi(e_1), \chi(e_2))$. The *reload cost* $c(P)$ of a path P of length ℓ with edges $e_1, e_2, \ldots, e_\ell$ is $c(P) := \sum_{i=2}^{\ell} c(e_{i-1}, e_i)$. Note that the reload cost of a path consisting of one edge is zero. Throughout this work, the terms distance, diameter and eccentricity will be used only with respect to this reload cost measure. The (reload cost) distance between two vertices u, v in a graph G is $\mathsf{dist}^c_G(u, v) = \min\{c(P) \mid P \text{ is a path between } u \text{ and } v \text{ in } G\}$. The (reload cost) *eccentricity* of a vertex u in a graph G is $\epsilon^c_G(u) = \max\{\mathsf{dist}^c_G(u, v) \mid v \in V(G)\}$. The (reload cost) *diameter* of a graph G is $\mathsf{diam}^c(G) := \max\{\epsilon^c_G(u) \mid u \in V(G)\}$. For notational convenience we assume that the edge-coloring function χ and the reload cost function c are clear from the context and omit the superscript c in the last three definitions.

The problem we study in this paper is defined as follows:

> Minimum Reload Cost Diameter Spanning Tree (Diameter-Tree)
> **Input:** A graph $G = (V, E)$ with an edge-coloring χ and a reload cost function c.
> **Output:** A spanning tree T of G minimizing $\mathsf{diam}(T)$.

If for every three distinct edges e_1, e_2, e_3 of G incident to the same node, it holds that $c(e_1, e_3) \leqslant c(e_1, e_2) + c(e_2, e_3)$, we say that the reload cost function c satisfies the *triangle inequality*. This assumption is sometimes used in practical applications [33].

Throughout the paper, we denote by n, Δ, and tw the number of vertices, the maximum degree, and the treewidth of the input graph, respectively. When we consider the (parameterized) decision version of the Diameter-Tree problem, we denote by k the desired cost of a solution.

Parameterized complexity. We refer the reader to [9, 13] for basic background on parameterized complexity, and we recall here only some basic definitions. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. The number k is termed the *parameter* of the instance $I = (x, k) \in \Sigma^* \times \mathbb{N}$.

An algorithm for a parameterized problem L is an algorithm that decides whether an input instance $I = (x, k)$ is in L. Such an algorithm is an FPT (resp. XP) algorithm if its running time is bounded by $f(k) \cdot |I|^c$ (resp. $f(k) \cdot |I|^{g(k)}$) for some computable functions f, g and some constant c. The class FPT (resp. XP) is the class of all parameterized problems for which an FPT (resp. XP) algorithm exists. Clearly, $\mathsf{FPT} \subseteq \mathsf{XP}$. A parameterized problem that is in FPT is termed *fixed-parameter tractable*. The Vertex Cover problem, parameterized by the size of the solution, is in FPT. The Clique problem, parameterized by the size of the solution, is in XP.

A parameterized problem is para-NP-*hard* if it is NP-hard when the value of its parameter is bounded by some *constant*. The Vertex Coloring problem parameterized by the number of colors is para-NP-hard. Note that, unless $\mathsf{P} = \mathsf{NP}$, a para-NP-hard problem cannot be in XP, hence it cannot be FPT either.

Within parameterized problems, the class $\mathsf{W}[1]$ may be seen as the parameterized equivalent to the class NP of classical optimization problems. Without entering into details (see [9,13] for the formal definitions), if a parameterized problem is $\mathsf{W}[1]$-*hard* then this problem is unlikely to be in FPT. The Clique problem parameterized by the size of the solution is the canonical example of a $\mathsf{W}[1]$-hard problem. To transfer $\mathsf{W}[1]$-hardness from one problem to another, one uses a *parameterized reduction*, which given an input $I = (x, k)$ of the source problem, computes an equivalent instance $I' = (x', k')$ of the target problem in time $f(k) \cdot |I|^c$, where f is a computable function, c is a constant, and k' is bounded by a function depending only on k.

Tree decompositions. A *tree decomposition* of a graph G is a pair $\mathcal{D} = (Y, \mathcal{X})$, where Y is a tree and $\mathcal{X} = \{X_t \mid t \in V(Y)\}$ is a collection of subsets of $V(G)$ such that:

- $\bigcup_{t \in V(Y)} X_t = V(G)$,

- for every edge $\{u, v\} \in E$, there is a $t \in V(Y)$ such that $\{u, v\} \subseteq X_t$, and
- for each $\{x, y, z\} \subseteq V(Y)$ such that z lies on the unique path between x and y in Y, $X_x \cap X_y \subseteq X_z$.

We call the vertices of Y *nodes* of $\mathcal{D}$ and the sets in $\mathcal{X}$ *bags* of $\mathcal{D}$. The width of the tree decomposition $\mathcal{D} = (Y, \mathcal{X})$ is $\max_{t \in V(Y)} |X_t| - 1$. The *treewidth* of G, denoted by $\mathsf{tw}(G)$, is the smallest integer w such that there exists a tree decomposition of G of width w.

Nice tree decompositions. Let $\mathcal{D} = (Y, \mathcal{X})$ be a tree decomposition of G, r be a vertex of Y designated as its root, and $\mathcal{G} = \{G_t \mid t \in V(Y)\}$ be a collection of subgraphs of G, indexed by the vertices of Y. A triple $(\mathcal{D}, r, \mathcal{G})$ is *nice* if:

- every node of $\mathcal{D}$ has at most two children in Y,
- for every leaf t of Y except r, $X_t = \varnothing$ and $G_t = (\varnothing, \varnothing)$ (t is termed a *leaf node* in this case),
- $X_r = \varnothing$ and $G_r = G$,
- if $t \in V(Y)$ has two children t' and t'', then $X_t = X_{t'} = X_{t''}$, and $E(G_{t'}) \cap E(G_{t''}) = \varnothing$ (t is termed a *join node* in this case),
- if $t \in V(\mathcal{T})$ has exactly one child t' one of the following holds:
  - $X_t = X_{t'} \cup \{v_{\text{insert}}\}$ for some $v_{\text{insert}} \notin X_{t'}$ and $G_t = (V(G_{t'}) \cup \{v_{\text{insert}}\}, E(G_{t'}))$ (t is termed a *vertex-introduce node* and v_{insert} is the *insertion vertex* of X_t),
  - $X_t = X_{t'}$ and $G_t = (G_{t'}, E(G_{t'}) \cup \{e_{\text{insert}}\})$ where e_{insert} is an edge of G with endpoints in X_t. In (t is termed an *edge-introduce node* and e_{insert} is the *insertion edge* of X_t),
  - $X_t = X_{t'} \backslash \{v_{\text{forget}}\}$ for some $v_{\text{forget}} \in X_{t'}$ and $G_t = G_{t'}$ (t is termed a *forget node* and v_{forget} is the *forget vertex* of X_t).

The notion of a nice triple defined above is essentially the same as the one of nice tree decomposition in [10] (which is in turn an enhancement of the original one, introduced in [28]). As already argued in [10, 28], it is possible to transform in polynomial time a given tree decomposition to a nice triple $(\mathcal{D}, r, \mathcal{G})$ such that $\mathcal{D}$ has the same width as the given tree decomposition.

Transfer triples and their fusion. For a graph G and a subset R of its vertices, denote by R^G the set of all edges and vertices of G except the edges incident to R, i.e., $R^G = V(G) \cup E(G) \backslash \mathsf{adj}_G(R)$. A triple (F, R, α) where F is a forest, $R \subseteq V(F)$, and $\alpha : R \times R^F \to [0, k] \cup \{\perp\}$ is a *transfer triple* if $\alpha(v, a) \neq \perp$ if and only if v and a are in the same connected component of F. See Figure 1 for an illustration. Intuitively, the function α will indicate the "cost of transferring" from v to a in F for each pair (v, a).

Let (F_1, R, α_1) and (F_2, R, α_2) be two transfer triples where F_1 and F_2 are edge-disjoint and their union F is a forest. Let also $\beta : \mathsf{adj}_{F_1}(R) \times \mathsf{adj}_{F_2}(R) \to [0, k] \cup \{\perp\}$.

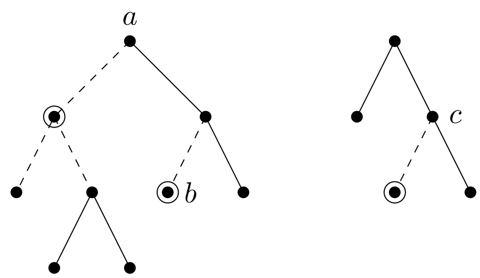

Figure 1: A transfer triple (F, R, α) where F is the depicted forest, R corresponds to the circled vertices, R^F is the set of all vertices and edges except the dashed edges, and α is such that $\alpha(a,b) \in [0,k]$ and $\alpha(a,c) = \bot$.

We require a function $\alpha_1 \oplus_\beta \alpha_2 : R \times R^F \to [0,k] \cup \{\bot\}$ that builds the transferring costs of moving in F by taking into account the corresponding transferring costs in F_1 and F_2. The values of $\alpha_1 \oplus_\beta \alpha_2$ are defined as follows:

Let $(v,a) \in R \times R^F$. Let P be the path in F between v and a and let $V(P) = \{v_0, \ldots, v_r\}$, ordered in the way these vertices appear in P and assuming that $v_0 = v$. To simplify notation, we assume that $\{v_0, v_1\}$ is an edge of F_1 (otherwise, exchange the roles of F_1 and F_2). Given $i \in [r-1]$, we define e_i^- (resp. e_i^+) as the edge incident to v_i that appears before (resp. after) v_i when traversing P from v to a. We define the set of indices

$$I = \{i \mid e_i^- \text{ and } e_i^+ \text{ belong to different sets of } \{E(F_1), E(F_2)\}\}.$$

Let $I = \{i_1, \ldots, i_q\}$, where numbers are ordered in increasing order and we also set $i_0 = 0$. Then we set

$$\begin{aligned} \alpha_1 \oplus_\beta \alpha_2(v,a) \quad = \quad & \sum_{h \in [0, \lfloor \frac{q-1}{2} \rfloor]} \alpha_1(v_{2i_h}, v_{2i_h+1}) + \sum_{h \in [0, \lfloor \frac{q-2}{2} \rfloor]} \alpha_2(v_{2i_h+1}, v_{2i_h+2}) \\ & + \sum_{h \in [q]} \beta(e_{i_h}^-, e_{i_h}^+) + \alpha_{(q \bmod 2)+1}(v_{i_q}, a). \end{aligned}$$

Roughly speaking, $\alpha_1 \oplus_\beta \alpha_2(v,a)$ is the cost of the path P from v to a in F calculated as the sum of the cost of each connected component, provided by α_1 and α_2, of $P \cap F_1$ and $P \cap F_2$ together with the sum of the costs, provided by β, of each transition from F_1 to F_2 and F_2 to F_1 used by P.

Satisfiability. An instance of 3-Sat is a boolean formula φ with n variables $x_1, \ldots, x_n$ and m clauses where each clause contains at most three literals, and a literal is an occurence of a variable or its negation. The goal is to decide whether there is a truth assignment of the variables that satisfies all the clauses. A clause is satisfied if and only if one of its literals is set to true by the assignment. We can assume without loss of generality that every variable occurs at least once positively and at least once negatively. Indeed, otherwise one can set the truth assignment of this variable appropriately and remove from the formula all the clauses in which this variable occurs.

The restriction of 3-Sat to formulas where each variable occurs in at most three clauses was proved to be NP-complete by Tovey [32]. It is worth mentioning that one needs to allow for clauses of size two or three, as if all clauses have size exactly three, then it turns out that all instances are satisfiable [32].

When working with this restriction, we may also assume:

✠ *Every variable occurs exactly three times in the formula φ.*

Indeed, let x be a variable occurring exactly two times in φ, once positively and once negatively. We obtain a new formula φ' from φ by adding a new variable y and two clauses $x \vee y$ and $y \vee \overline{y}$. Clearly φ and φ' are equivalent, and both x and y occur exactly three times in φ'. Applying these operations exhaustively results in an equivalent formula in which every variable occurs exactly three times.

3 Para-NP-hardness results

We start with the para-NP-hardness result for the parameter $k + \mathsf{tw}$.

Theorem 1. *The* Diameter-Tree *problem is* NP*-hard on outerplanar graphs with only one vertex of degree larger than 3, even with three different costs that satisfy the triangle inequality, and $k = 9$. Since outerplanar graphs have treewidth at most 2, in particular,* Diameter-Tree *is* para-NP*-hard when parameterized by $k + \mathsf{tw}$.*

Proof. We present a reduction from 3-Sat. Given a formula φ with n variables and m clauses, we create an instance (G, χ, c) of Diameter-Tree as follows. We may assume that no clause of φ contains a variable and its negation. Consult Figure 2 for the following construction. The graph G contains a distinguished vertex r and, and a clause gadget for every clause. The clause gadget C_j corresponding to the clause $c_j = (\ell_1 \vee \ell_2 \vee \ell_3)$, consists of three vertices $v^j_{\ell_1}, v^j_{\ell_2}, v^j_{\ell_3}$ and five edges $\{r, v^j_{\ell_1}\}$, $\{r, v^j_{\ell_2}\}$, $\{r, v^j_{\ell_3}\}$, $\{v^j_{\ell_1}, v^j_{\ell_2}\}$, and $\{v^j_{\ell_2}, v^j_{\ell_3}\}$. This completes the construction of G. Note that G does not depend on the formula φ except for the number of clause gadgets, and that it is an outerplanar graph with only one vertex of degree greater than 3, as required.

We proceed with the description of the coloring χ and the cost function c. For simplicity, we associate a distinct color with each edge of G, and thus, with slight abuse of notation, it is enough to describe the cost function c for every pair of incident edges of G, as we consider symmetric cost functions. We set

$$c(e_1, e_2) = \begin{cases} 10 & \text{if } e_1 = \{r, v^{j_1}_{\ell_{i_1}}\}, e_2 = \{r, v^{j_2}_{\ell_{i_2}}\} \text{ and } \ell_{i_1} = \overline{\ell_{i_2}}\ , \\ 5 & \text{if } e_1 = \{r, v^{j_1}_{\ell_{i_1}}\}, e_2 = \{r, v^{j_2}_{\ell_{i_2}}\} \text{ and } \ell_{i_1} \neq \overline{\ell_{i_2}}\ , \text{ and} \\ 1 & \text{otherwise.} \end{cases}$$

Note that this cost function satisfies the triangle inequality since the reload costs between edges incident to r are 5 and 10, and the reload costs between edges incident to other vertices are 1.

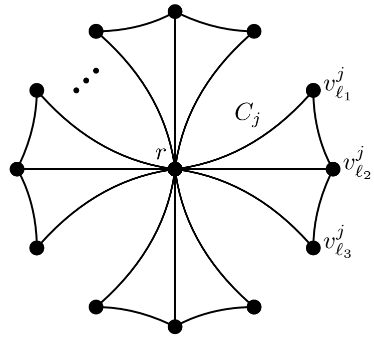

Figure 2: Example of the graph G built in the reduction of Theorem 1.

We claim that φ is satisfiable if and only if G contains a spanning tree with diameter at most 9. Since r is a cut vertex and every clause gadget is a connected component of $G - r$, in every spanning tree, the vertices of C_j together with r induce a tree with four vertices. Moreover, the reload cost of a path from r to any leaf of this tree is at most 2. Therefore, the diameter of any spanning tree is at most 4 plus the maximum reload cost incurred at r by a path of T.

Assume first that φ is satisfiable, and let ψ be a satisfying assignment of φ. We now construct a spanning tree T of G with $\mathsf{diam}(T) \leqslant 9$. For every clause c_j, the tree T^j is the tree spanning C_j and containing exactly one edge incident to r where the other endpoint of this edge is a literal of c_j that is set to true by ψ. T is the union of all the trees T_j constructed in this way. The reload cost incurred at r by any path of T traversing it is at most 5, since we never choose a literal and its negation. Therefore, $\mathsf{diam}(T) \leqslant 9$.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leqslant 9$. Then, the reload cost incurred at r by any path traversing it is at most 5 since otherwise $\mathsf{diam}(T) \geqslant 10$. For every $j \in [m]$, let T_j be the subtree of T induced by C_j and let $\{r, v^j_{\ell_{i_j}}\}$ be one of the edges incident to r in T_j. We note that for any pair of clauses c_{j_1}, c_{j_2} we have $\ell_{i_{j_1}} \neq \overline{\ell_{i_{j_2}}}$, since otherwise a path using these two edges would incur a cost of 10 at r. The variable in the literal ℓ_{i_j} is set by ψ so that ℓ_{i_j} is true. All the other variables are set to an arbitrary value by ψ. Note that ψ is well-defined, since we never encounter a literal and its negation during the assignment process. It follows that ψ is a satisfying assignment of φ. $\square$

We proceed with the $\mathsf{para\text{-}NP}$-hardness result for the parameter $k + \Delta$.

Theorem 2. *The* Diameter-Tree *problem is* NP*-hard on graphs with* $\Delta \leqslant 3$*, even with two different costs,* $k = 0$*, and a bounded number of colors. In particular, it is* $\mathsf{para\text{-}NP}$*-hard parameterized by* $k + \Delta$*.*

Proof. We present a reduction from the restriction of 3-Sat to formulae that satisfy property ✠. We recall that property ✠ states that each variable occurs exactly three times in the given formula φ of 3-Sat and each variable occurs at least once positively and at least once negatively in φ.

Given a formula φ with n variables and m clauses, we create an instance (G, χ, c) of Diameter-Tree with $\Delta(G) \leqslant 3$ as follows. Let $x_1, \ldots, x_n$ be the variables of φ. For every $i \in [n]$, G contains a *variable gadget* consisting of five vertices u_i, v_i, p_i, r_i, n_i and five edges $\{u_i, v_i\}, \{v_i, p_i\}, \{p_i, r_i\}, \{r_i, n_i\}, \{n_i, v_i\}$ (see Figure 3(a)). For every $i \in [n-1]$, G contains the edge $\{u_i, u_{i+1}\}$. G consists of a vertex c_j for every $j \in [m]$. For each variable x_i, the vertex p_i (resp. n_i) is connected to one of the vertices c_j corresponding to a clause of φ in which x_i appears positively (resp. negatively), and r_i is connected to the vertex corresponding to the remaining clause in which x_i appears (positively or negatively). This completes the construction of G. Note that $\Delta(G) \leqslant 3$ as required.

We now define the coloring χ and the cost function c. The color set X is $[9]$. For $i \in [n]$, $\chi(\{p_i, r_i\}) = 1$ and $\chi(\{r_i, n_i\}) = 2$, and all edges incident to u_i or v_i have color 3. For $j \in [m]$, we color the edges incident to c_j using distinct colors from $[4, 9]$. Edges corresponding to positive (resp. negative) occurrences get colors from $[4, 6]$ (resp. $[7, 9]$) (see Figure 3(b)). We set $c(1, 2) = 1$, $c(1, i) = 1$ for every $i \in [4, 6]$, $c(2, i) = 1$ for every $i \in [7, 9]$, and $c(i, j) = 1$ for every pair of distinct colors $i, j \in [4, 9]$. All other costs are set to 0.

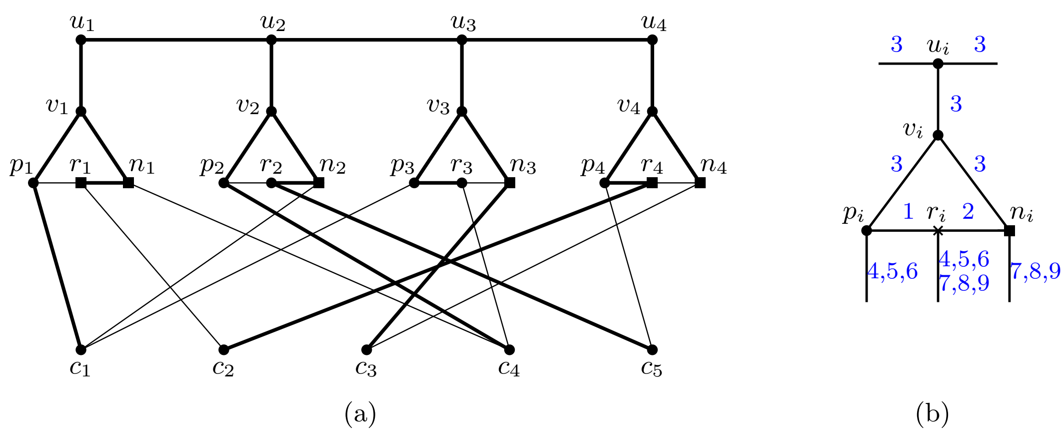

Figure 3: (a) Graph G described in the reduction of Theorem 2 for the formula $\varphi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee x_4)$. The vertices p_i, r_i, n_i corresponding to positive (resp. negative) occurrences are depicted with circles (resp. squares). An assignment satisfying φ is given by $x_1 = x_2 = 1$ and $x_3 = x_4 = 0$, and a solution spanning tree T with diameter 0 is emphasized with thicker edges. (b) The (possible) colors associated with each edge of G correspond to the (blue) numbers.

We now show that φ is satisfiable if and only if G contains a spanning tree with reload cost diameter 0. Assume first that φ is satisfiable, and let ψ be a satisfying assignment of φ. We construct a spanning tree T of G with diameter 0 as follows. For every $i \in [n]$, the tree T contains all the edges incident to u_i and v_i. If variable x_i is set to true (resp.

false) by ψ, T contains the edge $\{r_i, n_i\}$ (resp. $\{p_i, r_i\}$). For $j \in [m]$, the tree T contains one of the edges incident c_j that corresponds to a literal satisfying the j-th clause. It can be easily checked that T is a spanning tree of G with $\mathsf{diam}(T) \leqslant 0$ (see Figure 3(a)).

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) = 0$. Since the cost associated with any pair of distinct colors from $[4, 9]$ is 1, it follows that c_j is a leaf of T, for $j \in [m]$. Therefore, the variable gadgets need to be connected in T via the vertices u_i, implying that all the edges incident to u_i belong to T for every $i \in [n]$. Furthermore, T contains exactly three out of the four edges of the 4-cycle induced by $\{v_i, p_i, r_i, n_i\}$. Since $c(1, 2) = 1$ and $\mathsf{diam}(T) = 0$, the missing edge is either $\{p_i, r_i\}$ or $\{r_i, n_i\}$. We define an assignment ψ of the variables $x_1, \ldots, x_n$ as follows. The variable x_i is set to true by ψ if and only if the edge $\{r_i, n_i\}$ belongs to T. We claim that ψ satisfies φ. Indeed, let c_j be a vertex in G. Since c_j is a leaf of T, it is connected exactly to one of the vertices p_i, r_i, n_i for some $i \in [n]$. Suppose that the edge e incident to c_j in T corresponds to a positive occurrence of x_i, the other case being symmetric. Then, e is one of $\{c_j, p_i\}, \{c_j, r_i\}$. In both cases, if the edge $\{p_i, r_i\}$ were in T, this edge together with $\{c_j, p_i\}$ or $\{c_j, r_i\}$ would incur a cost of 1 in T, contradicting the hypothesis that $\mathsf{diam}(T) = 0$. Therefore, the edge $\{p_i, r_i\}$ cannot be in T, implying that the edge $\{r_i, n_i\}$ must be in T. According to the definition of the assignment ψ, this implies that variable x_i is set to true in ψ, and therefore the c-th clause of φ is satisfied by the variable x_i. □

Note that in the above reduction the cost function c does *not* satisfy the triangle inequality at vertices p_i or n_i for $i \in [n]$, and recall that this is unavoidable since otherwise the problem would be polynomial [33]. It is worth mentioning that using the ideas in the proof of [22, Theorem 4 of the full version] it can be proved that the Diameter-Tree problem is also NP-hard on *planar* graphs with $\Delta \leqslant 4$, $k = 0$, and a bounded number of colors; we omit the details here.

Finally, we present the para-NP-hardness result for the parameter $\mathsf{tw} + \Delta$.

Theorem 3. *The* Diameter-Tree *problem is* NP*-hard on planar graphs with* $\mathsf{tw} \leqslant 3$ *and* $\Delta \leqslant 3$*. In particular, it is* para-NP*-hard parameterized by* $\mathsf{tw} + \Delta$.

Proof. We present a reduction from the Partition problem, which is a typical example of a *weakly* NP-complete problem [19]. An instance of Partition is a multiset $S = \{a_1, a_2, \ldots, a_n\}$ of n positive integers, and the objective is to decide whether S can be partitioned into two subsets S_1 and S_2 such that $\sum_{x \in S_1} x = \sum_{x \in S_2} x = \frac{B}{2}$ where $B = \sum_{x \in S} x$.

Given an instance $S = \{a_1, a_2, \ldots, a_n\}$ of Partition, we create an instance (G, χ, c) of Diameter-Tree as follows. The graph G contains a vertex r, called the root, and for every integer a_i where $i \in [n]$, we add to G six vertices $u_i, u'_i, m_i, m'_i, d_i, d'_i$ and seven edges $\{u_i, u'_i\}$, $\{m_i, m'_i\}$, $\{d_i, d'_i\}$, $\{u_i, m_i\}$, $\{u'_i, m'_i\}$, $\{m_i, d_i\}$, and $\{m'_i, d'_i\}$. We denote by H_i the subgraph induced by these six vertices and seven edges. We add the edges $\{r, u_1\}, \{r, d_1\}$ and, for $i \in [n-1]$, we add the edges $\{u'_i, u_{i+1}\}$ and $\{d'_i, d_{i+1}\}$. Let G' be the graph constructed so far. We then define G to be the graph obtained from two disjoint copies of G' by adding an edge between both roots. Note that G is a planar graph with $\Delta(G) = 3$ and $\mathsf{tw}(G) = 3$. (The claimed bound on the treewidth can be easily seen by building a *path* decomposition of G with consecutive bags of the form $\{u'_{i-1}, d'_{i-1}, u_i, d_i\}, \{u_i, d_i, m_i, u'_i\}, \{d_i, m_i, u'_i, m'_i\}, \{d_i, u'_i, m'_i, d'_i\}, \ldots$.)

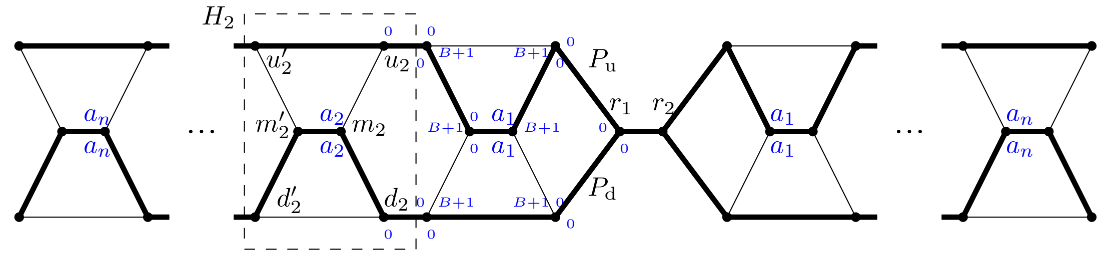

Figure 4: Graph G built in the reduction of Theorem 3, where the reload costs are depicted (in blue) at the angle between the two corresponding edges. For better visibility, not all costs and vertex labels are depicted. The typical shape of a solution spanning tree is highlighted with thicker edges.

Let us now define the coloring χ and the cost function c. Again, for simplicity, we associate a distinct color with each edge of G, and thus it is enough to describe the cost function c for every pair of incident edges of G. We define the costs for one of the copies of G', and the same costs apply to the other copy. For every edge e being either $\{u'_i, u_{i+1}\}$ or $\{d'_i, d_{i+1}\}$, for $1 \leqslant i \leqslant n-1$, we set $c(e, e') = 0$ for each of the four edges e' incident with e. For every edge $e = \{m_i, m'_i\}$, for $1 \leqslant i \leqslant n$, we set $c(\{u_i, m_i\}, e) = c(\{d_i, m_i\}, e) = a_i$ and $c(e, \{m'_i, u'_i\}) = c(e, \{m'_i, d'_i\}) = 0$. All costs associated with the two edges containing r in one of the copies G' are set to 0. For $e = \{r_1, r_2\}$, where r_1 and r_2 are the roots of the two copies of G', we set $c(e, e') = 0$ for each of the four edges e' incident to e. The cost associated with any other pair of edges of G is equal to $B + 1$; see Figure 4 for an illustration, where (some of) the reload costs are depicted (in blue), and a typical solution spanning tree of G is drawn with thicker edges.

We claim that the instance S of Partition is a Yes-instance if and only if G has a spanning tree with diameter at most B.

Assume first that S is a Yes-instance of Partition, and let $S_1, S_2 \subseteq S$ be a solution. We define a spanning tree T of G with diameter B as follows. We describe the subtree of T restricted to one of the copies of G', say T'. The spanning tree T of G is defined by union of two symmetric copies of T', one in each copy of G', together with the edge $\{r_1, r_2\}$. Tree T' consists of the two edges $\{r, u_1\}, \{r, d_1\}$ and two paths $P_{\mathrm{u}}, P_{\mathrm{d}}$ (corresponding to the upper and the lower path, respectively defined as follows; see Figure 4). For $i \in [n-1]$, the path P_{u} (resp. P_{d}) contains the edge $\{u'_i, u_{i+1}\}$ (resp. $\{d'_i, d_{i+1}\}$), and if $a_i \in S_1$ we add the three edges $\{u_i, m_i\}, \{m_i, m'_i\}, \{m'_i, u'_i\}$ to P_{u}, and the edge $\{d_i, d'_i\}$ to P_{d}. Otherwise, if $a_i \in S_2$, we add the edge $\{u_i, u'_i\}$ to P_{u} and the three edges $\{d_i, m_i\}, \{m_i, m'_i\}, \{m'_i, d'_i\}$ to P_{d}. Since $\sum_{x \in S_1} x = \sum_{x \in S_2} x = \frac{B}{2}$, it can be easily checked that both paths P_{u} and P_{d} have diameter $\frac{B}{2}$ in each of the two copies of G', and therefore T is a spanning tree of G with diameter B.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leqslant B$. Let G_1, G_2 be the two copies of G' in G, and let r_1, r_2 be their respective roots. Since the edge $\{r_1, r_2\}$ is a bridge of G, it necessarily belongs to T. By the construction of G, the choice of the reload costs, and since $\mathsf{diam}(T) \leqslant B$, it can be verified that, for $j \in \{1, 2\}$, $T \cap G_j$ consists of two paths $P_{\mathrm{u}}^j, P_{\mathrm{d}}^j$ intersecting at the root r_i where P_{u}^j (resp. P_{d}^j) contains all vertices u_i' and u_{i+1} (resp. d_i' and d_{i+1}) of the corresponding copy of G'. This can be observed by the fact that given $i_0 \in [n]$ the subpath $u_{i_0} m_{i_0} m_{i_0}' d_{i_0}'$ cannot be in T. First note that, as $c(\{u_{i_0}', m_{i_0}'\}, \{d_{i_0}', m_{i_0}'\}) = c(\{u_{i_0}', u_{i_0}\}, \{u_{i_0}, m_{i_0}\}) = B + 1 > B$, neither $\{u_{i_0}', m_{i_0}'\}$ nor $\{u_{i_0}', u_{i_0}\}$ is in T. As u_{i_0}' still needs to be covered by T, this implies that there exists $i > i_0$ such that in H_i, T contains a path from d_i to u_i but by construction, this implies that T should contain both $\{d_i, m_i\}$ and $\{m_i, u_i\}$ or both $\{d_i, d_i'\}$ and $\{d_i', m_i'\}$ or both $\{m_i', u_i'\}$ and $\{u_i', u_i\}$, three cases that imply the reload cost to be at least $B + 1$.

Furthermore, P_{u}^j (resp. P_{d}^j) contains the edge $\{u_i', u_{i+1}\}$ (resp. $\{d_i', d_{i+1}\}$) of the corresponding copy of G', and the intersection of P_{u}^j (resp. P_{d}^j) with the subgraph H_i in the corresponding copy of G' is given by either the three edges $\{u_i, m_i\}, \{m_i, m_i'\}, \{m_i', u_i'\}$ (resp. $\{d_i, m_i\}, \{m_i, m_i'\}, \{m_i', d_i'\}$) or by the edge $\{u_i, u_i'\}$ (resp. $\{d_i, d_i'\}$). Therefore, for $j \in \{1, 2\}$ and $x \in \{\mathrm{u}, \mathrm{d}\}$, it holds that $d_x^j := \mathsf{diam}(P_x^j) = \sum_{i \in I_x^j} a_i$, where I_x^j is the set of indices $i \in \{1, \ldots, n\}$ such that the edge $\{m_i, m_i'\}$ belongs to path P_x^j. Note also that, for $j \in \{1, 2\}$, by construction we have that $d_{\mathrm{u}}^j + d_{\mathrm{d}}^j = \sum_{i=1}^n a_i$, implying in particular that $\max\{d_{\mathrm{u}}^j, d_{\mathrm{d}}^j\} \geqslant \frac{B}{2}$. On the other hand, by the structure of T it holds that

$$B \geqslant \mathsf{diam}(T) \geqslant \max\{d_{\mathrm{u}}^1, d_{\mathrm{d}}^1\} + \max\{d_{\mathrm{u}}^2, d_{\mathrm{d}}^2\} \geqslant \frac{B}{2} + \frac{B}{2} = B. \tag{1}$$

Equation (1) implies, in particular, that $d_{\mathrm{u}}^1 = d_{\mathrm{d}}^1 = \frac{B}{2}$. In other words, $\sum_{i \in I_{\mathrm{u}}^1} a_i = \sum_{i \in I_{\mathrm{d}}^1} a_i = \frac{B}{2}$, thus the sets $I_{\mathrm{u}}^1, I_{\mathrm{d}}^1$ define a solution of Partition. This completes the proof. □

4 A polynomial-time algorithm for cactus graphs

In this section we present a polynomial-time algorithm for Diameter-Tree problem on cactus graphs. We begin with a definition of cactus graphs and an overview of their structure.

A *biconnected component*, or *block*, of a graph is a maximal biconnected induced subgraph of it. The *block tree* of a graph G is a tree whose nodes are the cut vertices and the blocks of G. Every cut vertex of G is adjacent in the block tree to all the blocks that contain it. The block tree of a graph is unique and can be computed in polynomial time [12]. A graph is a *cactus* graph if every block of it is either a cycle or a single edge. We term these blocks *cycle blocks* and *edge blocks*, respectively.

In this section, we present a polynomial-time algorithm that solves the decision version of the Diameter-Tree optimization problem, which we call Diameter-Tree*. Specifically, the input to the latter problem is an edge-colored graph G, a reload cost function c, and an integer k. The goal is to decide whether the input graph G has a spanning tree T with $\mathsf{diam}(T) \leqslant k$.

Observation 1. *The* Diameter-Tree *problem can be solved in time $\mathcal{O}(|I| \cdot f(|I|))$ if its decision version* Diameter-Tree* *can be solved in time $f(|I|)$, where $|I|$ denotes the size of the instance.*

Proof. Given an algorithm to decide Diameter-Tree*, one can perform a binary search over all the possible values for $\mathsf{diam}(T)$ to determine the smallest value k for which the algorithm returns Yes. Since $0 \leqslant \mathsf{diam}(T) < |V(G)| \cdot \max c$, this requires at most $\log(|V(G)| \cdot \max c = \log |V(G)| + \log \max c = \mathcal{O}(|I|)$ invocations of the decision algorithm. □

Our algorithm uses dynamic programming on the block tree of the input graph. We add a pendant vertex r to the input graph, so that its incident edge constitutes an edge block B_r. We color this edge with a new color, such that the reload cost of this color and any other color is zero. Clearly, the obtained instance is equivalent to the original. The algorithm processes the block tree of G in a bottom-up manner starting from its leaves, proceeding towards B_r while maintaining partial solutions for each block. At each step of the processing, it uses an algorithm that solves an instance of the 2-Sat problem as a subroutine. We proceed with definitions related to this structure.

We consider the block tree as a tree rooted at B_r. For a block B of G, we denote by $\mathcal{C}(B)$ the set of blocks that are immediate descendants of B in the block tree and refer to them as the *children* of B. Similarly, the *parent* of a block $B \neq B_r$ is the first block after B on the path from B to B_r in the block tree. We denote by $\mathcal{C}_E(B)$ and $\mathcal{C}_C(B)$ the sets of edge blocks and cycle blocks of $\mathcal{C}(B)$, respectively. We denote by G_B the subgraph of G induced by the union of all (not necessarily proper) descendants of B. The *anchor* $a(B)$ of a block $B \neq B_r$ is the cut vertex separating B from its parent, and $a(B_r) = r$.

Consult Figure 5 for the following definitions. Let B be a cycle block, and e an edge of B. Clearly, $B - e$ is a path P_e that contains all the vertices of B and the graph $G_B - e$ is connected. The vertex $a(B)$ divides P_e into two subpaths that we denote by $P_{e,+}$ and $P_{e,-}$ where the signs are chosen arbitrarily. Note that one of these subpaths is possibly trivial (i.e., it consists of one vertex, namely $a(B)$). We divide G_B into two induced subgraphs: subgraph $G_{B,e,+}$ (resp. $G_{B,e,-}$) is induced by the vertices that are reachable from $a(B)$ without using edges from $P_{e,-}$ (resp. $P_{e,+}$). Note that $G_{B,e,+}$ and $G_{B,e,-}$ intersect exactly at $a(B)$, and the degree of $a(B)$ is one in both subgraphs unless the subgraph under consideration is trivial. We denote by $a^+(B)$ (resp. $a^-(B)$) the unique neighbor of $a(B)$ in $G_{B,e,+}$ (resp. $G_{B,e,-}$). For $D \in \{+,-\}$, we denote $\mathcal{C}_{e,D}(B) = \{C \in \mathcal{C}(B) \mid a(C) \in P_{e,D}\}$. We have $G_{B,e,D} = P_{e,+} \cup \left(\bigcup_{C \in \mathcal{C}_{e,D}(B)} G_C\right)$. We define the set of graphs defined in this way by a block B as follows.

$$\mathcal{R}(B) = \begin{cases} \{G_B\} & \text{if } B \text{ is an edge block} \\ \{G_{B,e,+}, G_{B,e,-} \mid e \in E(B)\} & \text{if } B \text{ is a cycle block.} \end{cases}$$

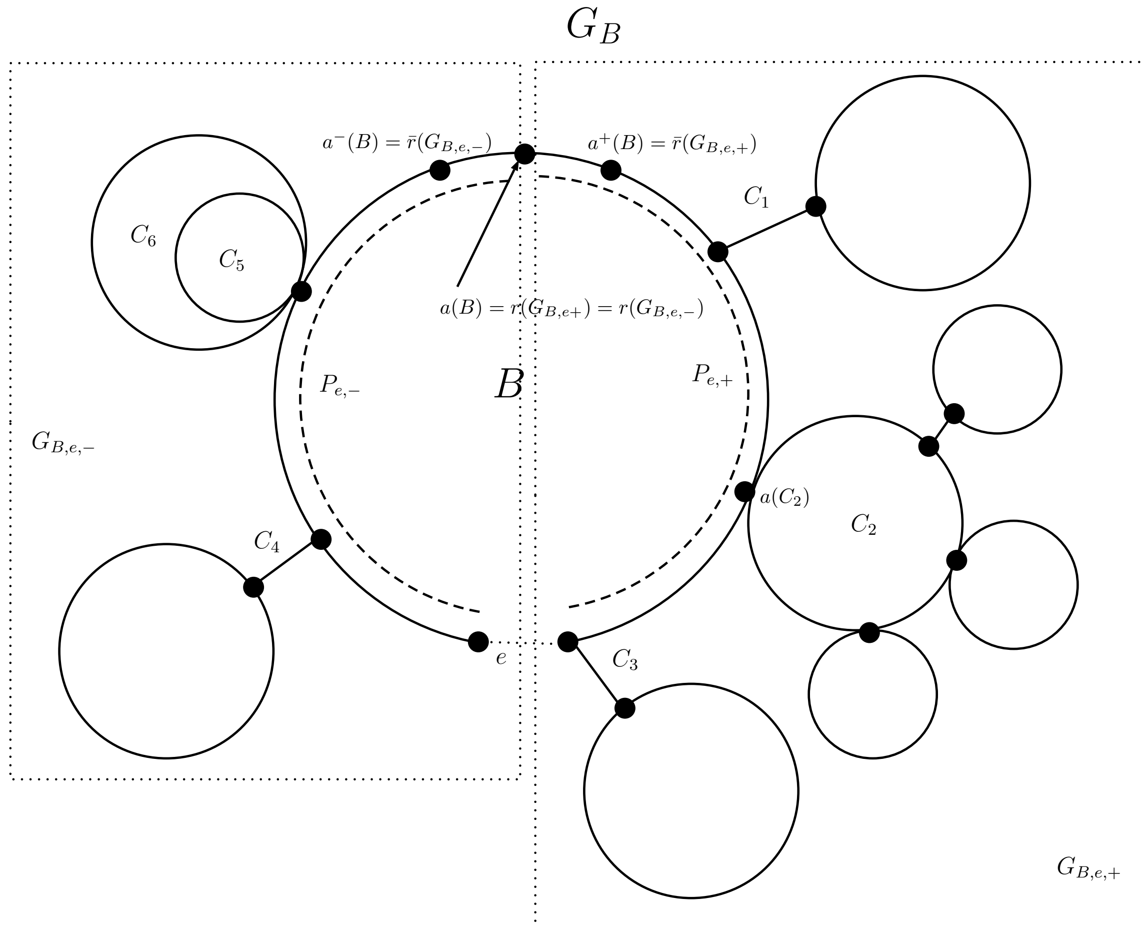

Figure 5: The subgraph G_B of a cactus G, defined by a cycle block B, and the two subgraphs defined by an edge e of B. The children of B are $\mathcal{C}(B) = \mathcal{C}_{e,+}(B) \cup \mathcal{C}_{e,-}(B)$, where $\mathcal{C}_{e,+}(B) = \{C_1, C_2, C_3\}$ and $\mathcal{C}_{e,+}(B) = \{C_4, C_5, C_6\}$.

We note that every $H \in \mathcal{R}(B)$ contains the vertex $a(B)$ which we denote as $r(H)$. Furthermore, if H is not trivial $r(H)$ has exactly one neighbor in H which we denote as $\bar{r}(H)$. Note that $\bar{r}(H) \in \{a^+(B), a^-(B)\}$ whenever H is not trivial. We use the same notations also for spanning subgraphs H' of H, i.e., $r(H') = r(H)$ and $\bar{r}(H') = \bar{r}(H)$.

A spanning tree T of G is obtained from G by the removal of one edge from every cycle block of G. Let T be a spanning tree of G that does not contain the edge e of some cycle block B. The spanning tree $T[G_B]$ of G_B is the union of two spanning trees: a tree $T_{B,e,+}$ spanning $G_{B,e,+}$ and a tree $T_{B,e,-}$ spanning $G_{B,e,-}$ that intersect exactly at $a(B)$.

We proceed with definitions and results that relate the diameters and eccentricities of a spanning tree within a cycle block and its children. We denote by $\mathcal{E}(B) = \times_{C \in \mathcal{C}_C(B)} E(C)$ the set of vectors that contain one edge from each cycle block $C \in \mathcal{C}_C(B)$. A vector

$\mathbf{e} \in \mathcal{E}(B)$ defines the following set of non-trivial subgraphs of G_B

$$\mathcal{R}_{\mathbf{e}} = \{G_C \mid C \in \mathcal{C}_E(B)\} \cup \{H = G[C, e_C, D] \mid C \in \mathcal{C}_C(B), D \in \{+,-\}, |V(H)| > 1\}.$$

Namely, this set contains one subgraph for every edge block in $\mathcal{C}(B)$ and two subgraphs for every cycle block in $\mathcal{C}(B)$. Observe that $\mathcal{R}_{\mathbf{e}} \subseteq \bigcup_{C\in\mathcal{C}(B)} \mathcal{R}(C)$. We further partition the set $\mathcal{R}_{\mathbf{e}}$ of graphs into two sets $\mathcal{R}_{\mathbf{e},+}$ and $\mathcal{R}_{\mathbf{e},+}$, according to the path that contains the root of the graph. Namely, $\mathcal{R}_{\mathbf{e},D} = \{H \in \mathcal{R}_{\mathbf{e}} \mid r(H) \in P_{e,D}\}$, for $D \in \{+,-\}$.

In the following lemma, for a spanning subgraph H of $G_{B,e,D}$ we use the shorthand $\epsilon(H)$ for $\epsilon_H(r(H))$.

Lemma 1. *Let B be a cycle block, e an edge of B, $D \in \{+,-\}$, $\mathbf{e} \in \times_{C\in\mathcal{C}(B)} E(C)$, and T a spanning tree of $G_{B,e,D}$ that does not contain the edges of $\mathbf{e}$. Then*

$$\epsilon(T) = \max\left(\epsilon_{P_{e,D}}(a(B)), \max_{H\in\mathcal{R}_{\mathbf{e},D}} (f(H) + \epsilon(T[H]))\right), \tag{2}$$

and $\mathsf{diam}(T)$ *is the maximum of the following four values:*

$$\begin{aligned}
&\mathsf{diam}(P_{e,D}),\\
&\max_{H\in\mathcal{R}_{\mathbf{e},D}} \mathsf{diam}(H),\\
&\max_{H\in\mathcal{R}_{\mathbf{e},D}} (h(H) + \epsilon(T[H])), \textit{ and}\\
&\max_{H,H'\in\mathcal{R}_{\mathbf{e},D}} \big(g(H,H') + \epsilon(T[H]) + \epsilon(T[H'])\big),
\end{aligned} \tag{3}$$

where f, g, h are functions that depend only on $e, \mathbf{e}$, and subgraphs of $\mathcal{R}_{\mathbf{e}}$, but not on T.

Proof. Let P be a path of G and $P_1, \ldots, P_\ell$ be subpaths of P such that P_1 and P_ℓ contain one endpoint of P each, and every two consecutive subpaths $P_i, P_{i+1}, (i \in [\ell-1])$ have exactly one edge in common. Then $c(P) = \sum_{i=1}^{\ell} c(P_i)$. Let $H \in \mathcal{R}_{\mathbf{e}}$, and consider a vertex $v \neq r(H)$ of H. We have

$$\mathsf{dist}_T(a(B), v) = \mathsf{dist}_T(a(B), \bar{r}(H)) + \mathsf{dist}_{T[H]}(r(H), v).$$

We note that $\mathsf{dist}_T(a(B), \bar{r}(H))$ does not depend on T but only on e, $\mathbf{e}$, and H, i.e., it is a function f of H. Therefore,

$$\begin{aligned}
\max_{v\in V(H)-r(H)} \mathsf{dist}_T(a(B), v) &= f(H) + \epsilon(T[H])\\
\max_{v\in V(T)-V(P_{e,D})} \mathsf{dist}_T(a(B), v) &= \max_{H\in\mathcal{R}_{\mathbf{e},D}} (f(H) + \epsilon(T[H])).
\end{aligned}$$

Clearly, $\max_{v\in V(P_{e,D})} \mathsf{dist}_T(a(B), v) = \epsilon_{P_{e,D}}(a(B))$. Combining the last two lines, we obtain (2).

We proceed with the diameter of T. Let v_1, v_2 be two vertices in distinct subgraphs $H_1, H_1 \in \mathcal{R}_{\mathbf{e}}$. We have

$$\mathsf{dist}_T(v_1, v_2) = \mathsf{dist}_T(\bar{r}(H_1), \bar{r}(H_2)) + \mathsf{dist}_{T[H_1]}(v_1, r(H_1)) + \mathsf{dist}_{T[H_2]}(v_2, r(H_2))$$

where the first term is a function g of H_1 and H_2. We conclude similarly:

$$\max_{v_1 \in H_1, v_2 \in H_2} \mathsf{dist}_T(v_1, v_2) = g(H_1, H_2) + \epsilon(T[H_1]) + \epsilon(T[H_2]).$$

Let v_1 be a vertex of $P_{e,D}$ and v_2 be a vertex of a subgraph $H \in \mathcal{R}_{\mathbf{e},D}$. We have

$$\begin{aligned}
\mathsf{dist}_T(v_1, v_2) &= \mathsf{dist}_T(v_1, \bar{r}(H)) + \mathsf{dist}_{T[H]}(v_2, r(H)) \\
\max_{v_2 \in H} \mathsf{dist}_T(v_1, v_2) &= \mathsf{dist}_T(v_1, \bar{r}(H)) + \epsilon(T[H]) \\
\max_{v_1 \in P_{e,D}, v_2 \in H} \mathsf{dist}_T(v_1, v_2) &= \left(\max_{v_1 \in P_{e,D}} \mathsf{dist}_T(v_1, \bar{r}(H)) \right) + \epsilon(T[H])
\end{aligned}$$

where the first term is a function $h(H)$. Therefore,

$$\max_{v_1 \in B, v_2 \in H} \mathsf{dist}_T(v_1, v_2) = h(H) + \epsilon(T[H]).$$

Considering pairs of vertices from B and pairs of vertices that are in the same subgraph $H \in \mathcal{R}_{\mathbf{e},D}$ we conclude (3). ☐

We proceed with the description of the algorithm. Our algorithm computes a set of "best" spanning trees for every block B. For a graph $H \in \mathcal{R}(B)$ of some block B, $\lambda(H)$ is a spanning tree T of H with diameter at most k that minimizes the eccentricity of $r(H)$ (in T). Formally, let $\mathcal{S}_k(H)$ denote the set of spanning trees of H having diameter at most k. Then

$$\lambda(H) = \begin{cases} \perp & \text{if } \mathcal{S}_k(H) = \emptyset \\ \arg\min_{T \in \mathcal{S}_k(H)} \epsilon_T(r(H)) & \text{otherwise.} \end{cases}$$

For every block B and every graph $H \in \mathcal{R}(B)$ we compute $\lambda(G_B)$. If $\lambda(G_B) = \perp$ for some edge block B then G_B (thus G as well) does not contain a spanning tree of diameter at most k. In this case the algorithm stops and returns No. If $(\lambda(G_{B,e,+}) = \perp) \vee (\lambda(G_{B,e,-}) = \perp) \vee \mathsf{diam}(\lambda(G_{B,e,+}) \cup \lambda(G_{B,e,-})) > k$ for every edge e of B of a cycle block B, then G_B does not contain a spanning tree with diameter at most k and the algorithm returns No. Otherwise, the processing continues until finally B_r is processed successfully and the algorithm returns Yes, since $\lambda(G_{B_r}) = \lambda(G)$ is a spanning tree of G with diameter at most k.

In the sequel we will mostly confine ourselves to cycle blocks, and overlook the case of edge block which is, by far, simpler. We first show the following lemma stating that the above strategy is valid, i.e., for every $H \in \mathcal{R}(B)$, a valid value for $\lambda(H)$ can be computed from $\{\lambda(H') \mid H' \in \mathcal{R}(C), C \in \mathcal{C}(B)\}$. We denote by $\Lambda(H)$ the set of all spanning trees T of H that can be obtained from spanning trees $\lambda(H')$ of the children of B. Formally,

$$\Lambda(H) = \left\{ P_{e,D} \cup \left(\bigcup_{H' \in \mathcal{R}_{\mathbf{e},D}} \lambda(H') \right) \mid \mathbf{e} \in \mathcal{E}(B) \right\}.$$

Lemma 2. *Let B be a cycle block and $H \in \mathcal{R}(B)$, i.e., $H = G_{B,e,D}$ for some edge e of B and some $D \in \{+,-\}$. If $\lambda(H) \neq \perp$ then $\Lambda(H)$ contains a valid value for $\lambda(H)$.*

Proof. Let $\bar{T}$ be a valid value for $\lambda(H)$. Since $\bar{T}$ is a spanning tree of $G_{B,e,D}$ it does not contain the edge e. Moreover, it does not contain exactly one edge e_C from every cycle block $C \in \mathcal{C}_C(B)$. Let $\mathbf{e}$ be the vector of these edges. Then

$$\bar{T} = P_{e,D} \cup \left(\bigcup_{H' \in \mathcal{R}_{\mathbf{e},D}} \bar{T}[H'] \right).$$

Clearly, $\mathsf{diam}(\bar{T}[H']) \leqslant \mathsf{diam}(\bar{T}) \leqslant k$ and similarly, $\mathsf{diam}(P_{e,D}) \leqslant k$. We define the following spanning tree of H.

$$\bar{T}_\lambda = P_{e,D} \cup \left(\bigcup_{H' \in \mathcal{R}_{\mathbf{e},D}} \lambda(H') \right).$$

By definition, $\mathsf{diam}(\lambda(H')) \leqslant k$ and $\epsilon(\lambda(H')) \leqslant \epsilon(\bar{T}(H'))$ for every H'. Then, by Lemma 1, we have $\mathsf{diam}(\bar{T}_\lambda) \leqslant k$ and $\epsilon(\bar{T}_\lambda) \leqslant \epsilon(\bar{T})$, i.e., $\bar{T}_\lambda \in \Lambda(H)$ is a valid value for $\lambda(H)$. □

We now reduce the problem of computing $\lambda(H)$ to a decision problem. We define boolean functions $\bar{\lambda}$ as follows. For a block B and $H \in \mathcal{R}(B)$ the value $\bar{\lambda}(H,i)$ is true if and only if there exists a spanning tree T of H such that $\mathsf{diam}(T) \leqslant k$ and $\epsilon(T) \leqslant i$.

Observation 2. Diameter-Tree* *can be decided in time $|I|^2 \cdot f(|I|)$ if $\bar{\lambda}$ can be computed in time $f(|I|)$ where $|I|$ denotes the size of the instance.*

Proof. Given an algorithm $\mathcal{A}$ that computes the function $\bar{\lambda}$ in time $f(|I|)$, we can obtain an algorithm that computes the function λ in time $|I| \cdot f(|I|)$ as follows. Perform a binary search over the possible values of i (i.e., $[0,k]$) by invoking $\mathcal{A}(H,i)$ to find the smallest value of i for which $\mathcal{A}$ returns true. This requires at most $\log k$ invocations of $\mathcal{A}$ which is linear in the size of the input, as observed in Observation 1. Since the functions λ are computed once for every edge of G, the result follows. □

In the rest of this section we present a polynomial-time algorithm for the following decision problem

Input: A graph $H \in \mathcal{R}(B)$ where B is a cycle block,
a nonnegative integer $i \leqslant k$,
$\{\lambda(H') \mid H' \in \mathcal{R}(C), C \in \mathcal{C}(B)\}$.
Output: $\bar{\lambda}(H,i)$.

Let $H = G_{B,e,D}$ for some edge e of B and $D \in \{+,-\}$, We build an instance ϕ of 2-Sat over the following variables

$$\left\{x_{e',D} \mid e' \in C, C \in \mathcal{C}_C(B), D \in \{+,-\}\right\}.$$

A subgraph F of H implies the following truth assignment to these variables: $x_{e',D}$ is true if and only if F contains the path $P_{e',D}$. We denote by $\phi(F)$ the truth value of ϕ resulting from such an assignment.

We first construct a formula ϕ_1 such that $\phi_1(T)$ is true for any spanning tree T of H. Then we construct a formula ϕ_2 such that $\phi_2(T)$ is true for a spanning tree $T \in \Lambda(H)$ of H such that $\mathsf{diam}(T) \leqslant k$, $\epsilon(T) \leqslant i$.

To finalize the proof, we will show that ϕ is satisfiable if and only if $\bar{\lambda}(H,i)$ is true.

We start with the construction of ϕ_1. For two consecutive edges e_1, e_2 of a cycle block $C \in \mathcal{C}_C(B)$, where e_1 is in $P_{e_2,+}$ (therefore, e_2 is in $P_{e_1,-}$). the formula ϕ_1 contains two clauses $x_{e_2,+} \vee x_{e_1,-}$ and $\overline{x_{e_2,+}} \vee \overline{x_{e_1,-}}$. Every spanning tree T of H satisfies $x_{e_2,+} \vee x_{e_1,-}$ since otherwise T contains neither $P_{e_2,+}$ nor $P_{e_1,-}$ implying that the common vertex v of e_1 and e_2 is not reachable from $a(C)$ in T. Similarly T satisfies $\overline{x_{e_2,+}} \vee \overline{x_{e_1,-}}$ since otherwise T contains both $P_{e_2,+}$ and $P_{e_1,-}$ implying that v is reachable from $a(C)$ using two edge-disjoint paths.

We proceed with the construction of ϕ_2. Let

$$T^{(e)} = P_{e,D} \cup \left(\bigcup_{C \in (\mathcal{C}_E(B) \cap \mathcal{C}_{B,e,D})} \bigcup_{H' \in \mathcal{R}(C)} \lambda(H') \right),$$

i.e., the tree obtained by the union of $P_{e,D}$ and all the trees $\lambda(H')$ that intersect $P_{e,D}$. Note that every tree spanning H contains $T^{(e)}$ as a subtree. For $H' = G_{C,e',D'} \in \mathcal{R}(C)$ we denote by $x_{H'}$ the variable $x_{e',D'}$ that corresponds to H'. Also, we denote by $P_{H'}$ the path $P_{e',D'}$, which is the intersection of H' with C. For every pair of graphs $\lambda(H'_1), \lambda(H'_2)$ of the input such that $\mathsf{diam}(T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2)) > k$ or $\epsilon(T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2)) > i$ we add to ϕ_2 a clause $\overline{x_{H'_1}} \vee \overline{x_{H'_2}}$.

The following observation is important for the rest of this section.

Observation 3. *Let $H', H'' \in \mathcal{R}(C)$ be such that H' is a subgraph H''. Then $\epsilon(\lambda(H')) \leqslant \epsilon(\lambda(H''))$ for any valid values of $\lambda(H')$ and $\lambda(H'')$.*

Lemma 3. *If $\bar{\lambda}(H,i)$ is true then ϕ is satisfiable.*

Proof. By Lemma 2, $\Lambda(H)$ contains a spanning tree T^* that is a valid value for $\lambda(H)$. Since $\bar{\lambda}(H,i)$ is true, $\mathsf{diam}(T^*) \leqslant k$ and $\epsilon(T^*) \leqslant i$. We will show that $\phi(T^*)$ is true. We have already observed that $\phi_1(T^*)$ is true. It remains to show that $\phi_2(T^*)$ is true.

Assume for a contradiction that a clause $\overline{x_{H'_1}} \vee \overline{x_{H'_2}}$ of ϕ_2 is not satisfied by T^* for some $H'_1 \in \mathcal{R}(C_1)$ and some $H'_2 \in \mathcal{R}(C_2)$. By the construction of ϕ_2, $\mathsf{diam}(T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2)) > k$ or $\epsilon(T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2)) > i$. Since the clause is not satisfied, both $x_{H'_1}(T^*)$ and $x_{H'_2}(T^*)$ are true. Therefore, T^* contains both $P_{H'_1}$ and $P_{H'_2}$. For $j \in \{1,2\}$, let $P_{H''_j}$

be the longest path P_{e',C_j} that contains $P_{H'_j}$ and contained in T^*. Then T^* contains both $\lambda(H''_1)$ and $\lambda(H''_2)$. By Observation 3, we have $\epsilon(\lambda(H''_j)) \geqslant \epsilon(\lambda(H''_j))$ for $j \in \{1,2\}$. Then $\epsilon(T^*) \geqslant \epsilon(T^{(e)} \cup \lambda(H''_1) \cup \lambda(H''_2)) \geqslant \epsilon(T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2))$. Furthermore, by Lemma 1, $\mathsf{diam}(T^*) \geqslant \mathsf{diam}(T^{(e)} \cup \lambda(H''_1) \cup \lambda(H''_2)) \geqslant \mathsf{diam}(T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2))$. We conclude that either $\mathsf{diam}(T^*) > k$ or $\epsilon(T^*) > i$, a contradiction. $\square$

Lemma 4. *If ϕ is satisfiable then $\bar{\lambda}(H,i)$ is true.*

Proof. Consider the truth values of the variables x under an assignment that satisfies ϕ. Consider the graph $J = P_{e,D} \cup \left(\bigcup_{x_{e',D'}=true} P_{e',D'}\right)$. Since ϕ_1 is satisfied by the assignment, J is a spanning tree of the graph of G_B consisting of B and its child cycle blocks. In other words, J misses exactly one edge e_C from each cycle bock $C \in \mathcal{C}_C(B)$. Indeed, otherwise either a child block C is contained in J, in which case every vertex of C is reachable from $a(C)$ using two vertex disjoint paths, or two edges of a cycle block C are not in J, in which case there is a vertex of C that is not reachable from $a(C)$. In both cases ϕ_1 contains a falsified clause. Let $\mathbf{e} \in \mathcal{E}(B)$ be the vector consisting of the edges not contained in J, and let

$$T^* = T^{(e)} \cup \left(\bigcup_{C \in \mathcal{C}(B)} (\lambda(G_{C,e_C,+}) \cup \lambda(G_{C,e_C,-})) \right).$$

Clearly, $T^* \in \Lambda(G_{B,e,D})$. It remains to show that $\mathsf{diam}(T^*) \leqslant k$ and $\epsilon(T^*) \leqslant i$. Assume that $\mathsf{diam}(T^*) > k$. Then T^* contains a path P between two vertices of u, v with $c(P) > k$. This path intersects at most two subgraphs $\lambda(H'_1)$ and $\lambda(H'_2)$, i.e., P is a path of $T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2)$. Therefore, $\mathsf{diam}(T^{(e)} \cup \lambda(H'_1) \cup \lambda(H'_2)) > k$, implying that ϕ_2 contains the clause $\overline{x_{H'_1}} \vee \overline{x_{H'_2}}$. Since this ϕ_2 is satisfied, at least one of $x_{H'_1}, x_{H'_2}$, say $x_{H'_1}$,is false in the assignment. Then J (therefore also T^*) does not contain $P_{H'_1}$, contradicting the fact that T^* contains $\lambda(H'_1)$. $\square$

Corollary 1. *$\bar{\lambda}(H,i)$ can be computed in time $\mathcal{O}(n^2)$ where n is the number of vertices of G.*

Proof. The number of variables in ϕ is at most twice the number of edges of G which is linear in the number of vertices of G. Therefore, the size of ϕ is $\mathcal{O}(n^2)$. Using Lemma 1, the diameters and eccentricities of the graphs can be computed in constant time from the diameters and eccentricities of their subgraphs. Therefore, every clause of ϕ can be generated in constant time. Finally, the satisfiability of ϕ can be checked in time linear to its size, i.e., $\mathcal{O}(n^2)$. $\square$

Combining the above corollary with Observations 1 and 2 we conclude

Theorem 4. *The* Diameter-Tree *problem can be solved in time $\mathcal{O}(|I|^5)$ for cactus graphs where $|I|$ is the size of the instance.*

5 FPT algorithm parameterized by $k + \mathsf{tw} + \Delta$

In this section we prove that the Diameter-Tree problem is FPT on general graphs parameterized by k, tw, and Δ. The proof is based on standard, but nontrivial, dynamic programming on graphs of bounded treewidth. It should be mentioned that we can assume that a tree decomposition of the input graph G of width $\mathcal{O}(\mathsf{tw})$ is given together with the input. Indeed, by using for instance the algorithm of Bodlaender *et al.* [7], we can compute in time $2^{\mathcal{O}(\mathsf{tw})} \cdot n$ a tree decomposition of G of width at most $5\mathsf{tw}$. Note that this running time is clearly dominated by the running time stated in Theorem 5. Recall also that, as mentioned in Section 2, it is possible to transform in polynomial time a given tree decomposition to a nice triple $(\mathcal{D}, r, \mathcal{G})$ such that $\mathcal{D}$ has the same width as the given tree decomposition.

Theorem 5. *The* Diameter-Tree *problem can be solved in time $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot n^{\mathcal{O}(1)}$. In particular, it is* FPT *parameterized by $k + \mathsf{tw} + \Delta$.*

Proof. The general idea behind the following algorithm is, given a tree decomposition $D = (Y, \mathcal{X} = \{X_t \mid t \in V(Y)\})$ of a graph G, to keep track, at each step $t \in V(T)$ of the dynamic programming algorithm, of the solution tree, restricted to the already explored part of the graph, together with the distance function from vertices of $V(G_t)$ to vertices of X_t in this solution. In order to reduce the size of the dynamic programming table, we compress the partial solution tree. This is the role of the reduce function that is defined below. In particular, we show that the obtained reduced forest has size linear in $|X_t|$. The α function that will be introduced later keeps track of the mentioned distances. After explaining these concepts, it remains to explain how the algorithm combines partial solutions in order to obtain the ones of the next step.

In order to provide a formal description of the dynamic programming, we need some more definitions. Given a graph G and a set S of its vertices, we say that S is *good for G* if S intersects every connected component of G. Let F be a forest and S be a set of vertices of F that is good for F. We define $\mathsf{Reduce}(F, S)$ as the forest F' that is obtained from F by repetitively applying the following operations to vertices that are not in $N_F[S]$ as long as this is possible:

1. removing a vertex of degree 1 and
2. dissolving a vertex of degree 2.

Suppose now that $\mathsf{Reduce}(F, S) = F'$. We define the associated *reduce function* $\varphi : V(F) \to V(F') \cup E(F')$ as follows. For every vertex $z \in V(F)$, we define K_z to be the set of vertices x of $V(F')$ such that there exists a path in F from z to x that does not use any vertex of $V(F') \backslash \{x\}$. Note that, if $z \in V(F')$, then $K_z = \{z\}$, since for each $x \in V(F') \backslash \{z\}$, the path from z to x contains z, which is an element of $V(F') \backslash \{x\}$. If K_z contains only one element x, then we define $\varphi(z) = x$, otherwise we define $\varphi(z) = K_z$. To show that φ is well-defined, we claim that $1 \leqslant |K_z| \leqslant 2$ and if $|K_z| = 2$ then $K_z \in E(F')$. Indeed, since each connected component of F contains an element of S, we have $|K_z| \geqslant 1$.

Assume that K_z contains two distinct vertices x_1 and x_2. By definition, we know that x_1 and x_2 are in the same connected component of F and also of F'. Let P_i be the path from z to x_i, $i \in \{1, 2\}$, in F and let P be the path from x_1 to x_2 in $F[V(P_1) \cup V(P_2)]$. By definition of x_1 and x_2, $V(P) \cap V(F') = \{x_1, x_2\}$. Moreover, since F is a forest, P is the unique x_1x_2-path in F. Let us assume that $\{x_1, x_2\}$ is not an edge of F' and let x_3 be a vertex of F' on the path from x_1 to x_2 in F'. Then x_3 should be in P. This contradicts the fact that $V(P) \cap V(F') = \{x_1, x_2\}$. As F' is a forest, this also implies that $|K_z| \leqslant 2$. Intuitively, given $z \in V(F)$, either $z \in V(F')$ and so $K_z = \{z\}$, or z has been removed from the forest after the application of Operation 1 or Operation 2 and then K_z corresponds to the vertex or edge we should start from if we want to recover z using the reverse operation of Operation 1 and Operation 2. Roughly speaking, a vertex $z \in V(F)$ will be represented by a vertex or an edge of F', depending on K_z, thanks to the function φ. Note that a given vertex or edge in F' can represent 0, 1, or more vertices of F.

We now proceed with the dynamic programming algorithm that solves DIAMETER-TREE*, the decision version of DIAMETER-TREE. Let (G, χ, c, k) be an instance of DIAMETER-TREE*. Consider a nice triple $(\mathcal{D}, r, \mathcal{G})$ where $\mathcal{D}$ is a tree decomposition $D = (Y, \mathcal{X} = \{X_t \mid t \in V(Y)\})$ of G with width at most tw and $\mathcal{G} = \{G_t \mid t \in V(Y)\}$. For each $t \in V(Y)$ we set $w_t = |X_t|$ and $V_t = V(G_t)$. We also refer to the vertices of X_t as *t-terminals* and to the edges that are incident to vertices in X_t as *t-terminal edges*. Given a forest F such that $X_t \subseteq V(F)$, we denote by X_t^F the set of all t-terminals and all non-t-terminal edges of F.

We provide a table $\mathcal{R}_t$ that the dynamic programming algorithm computes for each node of $\mathcal{D}$. For this, we need first the notion of a *t-pair*, that is a pair (F, α) where:

- F is a forest such that
  1. X_t is good for F,
  2. $X_t \subseteq V(F)$,
  3. $N_F(X_t) \subseteq N_G(X_t)$,
  4. $|V(F) \backslash N_F[X_t]| \leqslant w_t - 2$, and
  5. $|\{e \in E(F) \mid e \cap X_t = \varnothing\}| \leqslant 2w_t - 3$ and
- $\alpha : X_t \times X_t^F \to [0, k] \cup \{\bot\}$ is such that

  $\alpha(v, a) \neq \bot$ if and only if v and a are in the same connected component of F.

We call the vertices in $V(F) \backslash N_F[X_t]$ *external* vertices of F and the edges of $\{e \in E(F) \mid e \cap X_t = \varnothing\}$ *external* edges of F. Intuitively a t-pair (F, α), corresponding to a partial solution, is such that there exists a spanning forest $\hat{F}$ of G_t, such that $F = \mathsf{Reduce}(\hat{F}, X_t)$ and for each $z \in V(\hat{F})$ and $v \in X_t$, the cost to go from z to v in $\hat{F}$ is upper-bounded by $\alpha(v, K_z)$. The reason why we do not allow the edges incident to X_t to be modified, by the reduce function, is because we will need them to compute the reload cost when we will consider the next bags.

We need the function $\beta_t : \binom{\mathsf{adj}_G(X_t)}{2} \to [0,k] \cup \{\perp\}$ so that, for each $e_1, e_2 \in \mathsf{adj}_G(X_t)$, if there exists $x \in X_t$ such that $e_1 \cap e_2 = \{x\}$, then $\beta_t(e_1, e_2) = c(e_1, e_2)$, otherwise $\beta_t(e_1, e_2) = \perp$. Intuitively, the function β_t takes the role of the reload cost function but is defined between edges incident to a given vertex of the working bag X_t instead of being defined between colors. In particular, thanks to this, we do not have to take into consideration the number of colors.

Let (F, α) be a t-pair. Given a t-pair (F, α) as above we say that it is *admissible* if for every $(a, a') \in X_t^F \times X_t^F$ one of the following holds:

- there is no path between a and a' in F containing a vertex in X_t,
- one, say a, of a, a' is a vertex in X_t and $\alpha(a, a') \leqslant k$,
- some internal vertex b of the path P between a and a' in F belongs in X_t and $\alpha_t(b, a) + \beta_t(e^-, e^+) + \alpha_t(b, a') \leqslant k$, where e^+, e^- are the two edges in P that are incident to b.

Intuitively, the admissibility of a t-pair (F, α) assures that the transferring cost, indicated by α, between any two external elements is bounded by k.

We also need to explain how to combine t-pairs. Let (F_1, α_1) and (F_2, α_2) be two t-pairs where F_1 and F_2 are edge-disjoint and their union F is a forest. Let also $\beta : \mathsf{adj}_{F_1}(X_t) \times \mathsf{adj}_{F_2}(X_t) \to [0,k] \cup \{\perp\}$. We define the function $\alpha_1 \oplus_\beta \alpha_2 : X_t \times X_t^F \to [0,k] \cup \{\perp\}$ that builds the transferring costs of moving in F by taking into account the corresponding transferring costs in F_1 and F_2. The values of $\alpha_1 \oplus_\beta \alpha_2$ are defined as follows. Let $(v, a) \in X_t \times X_t^F$. Let P be the path in F between v and a and let $V(P) = \{v_0, \ldots, v_r\}$, ordered in the way these vertices appear in P and assuming that $v_0 = v$. To simplify notation, we assume that $\{v_0, v_1\}$ is an edge of F_1 (otherwise, exchange the roles of F_1 and F_2). Given $i \in [r-1]$, we define e_i^- (resp. e_i^+) as the edge incident to v_i that appears before (resp. after) v_i when traversing P from v to a. We define the set of indices

$$I = \{i \mid e_i^- \text{ and } e_i^+ \text{ belong to different sets of } \{E(F_1), E(F_2)\}\}.$$

Let $I = \{i_1, \ldots, i_q\}$, where numbers are ordered in increasing order and we also set $i_0 = 0$. Then we set

$$\begin{aligned} \alpha_1 \oplus_\beta \alpha_2(v, a) \quad = \quad & \sum_{h \in [0, \lfloor \frac{q-1}{2} \rfloor]} \alpha_1(v_{2i_h}, v_{2i_h+1}) + \sum_{h \in [0, \lfloor \frac{q-2}{2} \rfloor]} \alpha_2(v_{2i_h+1}, v_{2i_h+2}) \\ & + \sum_{h \in [q]} \beta(e_{i_h}^-, e_{i_h}^+) + \alpha_{(q \bmod 2)+1}(v_{i_q}, a). \end{aligned}$$

Roughly speaking, $\alpha_1 \oplus_\beta \alpha_2(v, a)$ is the cost of the path P from v to a in F calculated as the sum of the cost of each connected component, provided by α_1 and α_2, of $P \cap F_1$ and $P \cap F_2$ together with the sum of the costs, provided by β, of each transition from F_1 to F_2 and F_2 to F_1 used by P.

It is now time to give the precise definition of the table $\mathcal{R}_t$ of our dynamic programming algorithm. A pair (F, α) belongs in $\mathcal{R}_t$ if G contains a spanning tree $\hat{T}$ where $\mathsf{diam}(T) \leqslant k$ and the forest $\hat{F} = \hat{T}[V_t]$ (i.e., the restriction of $\hat{T}$ to the part of the graph that has been processed so far) satisfies the following properties:

- $\mathsf{Reduce}(\hat{F}, X_t) = F$, with the reduce function φ,
- for each $x \in X_t$ and $y \in X_t^F$, $\alpha(x, y) = \perp$ if and only if x and y are in two different connected components in F and if $\alpha(x, y) \neq \perp$, then for each $z \in \varphi^{-1}(y)$, $\mathtt{cost}_{\hat{F}}(x, z) \neq \perp$ and $\alpha(x, y) \geqslant \mathtt{cost}_{\hat{F}}(x, z)$.

Notice that each (F, α) as above is a t-pair. Indeed, Conditions 1–3 follow by the fact that $\hat{T}$ is a spanning tree of G and therefore $\hat{F}$ is a spanning forest of G_t. Conditions 4 and 5 follow by the fact that the number of internal vertices (resp. edges) of a tree with no vertices of degree two is at most two less than the number of leaves (resp. at most twice the number of leaves minus three). Moreover, the values of α are bounded by k because the diameter of $\hat{T}$ is at most k and therefore the same holds for all the connected components of $\hat{F}$. Notice that, for the same reason, all pairs in $\mathcal{R}_t$ must be admissible.

In the above definition, the external vertices and edges of F correspond to the parts of $\hat{F}$ that have been "compressed" during the reduction operation and the function α stores the transfer costs between those parts and the terminals. In this way, the trees in the t-pairs in $\mathcal{R}_t$ "represent" the restriction of all possible solutions in G_t. Moreover, the values of α indicate how these partial solutions interact with the t-terminals.

Our next concern is to bound the size of $\mathcal{R}_t$.

Claim 1. *For every $t \in V(Y)$, it holds that $|\mathcal{R}_t| \leqslant k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$.*

Proof. As we impose $N[X_t] \subseteq V(F)$, we have at most $2^{\Delta \cdot \mathsf{tw}}$ choices for the set $\{e \in E(F) \mid e \cap X_t \neq \emptyset\}$ and at most $(\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$ choices for the other edges or vertices. So the number of forests we take into consideration in $\mathcal{R}_t$ is at most $2^{\Delta \cdot \mathsf{tw}} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$. As the number of vertices and the number of edges of F is upper bounded by $\mathcal{O}(\Delta \cdot \mathsf{tw})$, the number of functions α is at most $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)}$. So $|\mathcal{R}_t| \leqslant k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$ and the claim holds. $\square$

Clearly, (G, χ, c, k) is a YES-instance if and only if $\mathcal{R}_r \neq \emptyset$. We now proceed with the description of how to compute the set $\mathcal{R}_t$ for every node $t \in \mathcal{T}$. For this, we will assume inductively that, for every descendant t' of t, the set $\mathcal{R}_{t'}$ has already been computed. We distinguish several cases depending on the type of node t:

- If t is a *leaf node*. Then $G_t = \{\emptyset, \emptyset\}$ and $\mathcal{R}_t = \{((\emptyset, \emptyset), \emptyset)\}$.
- If t is an *vertex-introduce node*. Let v be the insertion vertex of X_t and let t' be the child of t. Then

$$\begin{aligned} R_t \quad = \quad & \big\{\big((V(F') \cup \{v\}, E(F')), \alpha\big) \mid \exists (F', \alpha') \in R_{t'} : \\ & \qquad \alpha = \alpha' \cup \big\{\big((v, v), 0\big)\big\} \cup \big\{\big((v, a), \perp\big) \mid a \in X_t^{F'} \setminus \{v\})\big)\big]\big\}. \end{aligned}$$

Notice that at this point v is just an isolated vertex of G_t. This vertex is added in F and α is updated with the corresponding "void" transfer costs.

- If t is an *edge-introduce node*. Let $e = \{x, y\}$ be the insertion edge of X_t and let t' be the child of t. We define $F'' = (X_t, \{e\})$ and we set up $\alpha'' : X_t \times X_t^{F''} \to [0, k] \cup \{\perp\}$ (notice that $X_t^{F''} = X_t$) so that $\alpha''(x, y) = \alpha''(y, x) = 0$ and is $\perp$ for all other pairs of $X_t \times X_t$. Then

$$\begin{aligned} R_t \quad = \quad & R_{t'} \cup \{(F, \alpha) \mid (F, \alpha) \text{ is admissible, } F \text{ is a forest, and there exists a pair} \\ & (F', \alpha') \in R_{t'} \text{ such that } F = F' \cup F'' \text{ and } \alpha = \alpha' \oplus_{\beta_t} \alpha''\}. \end{aligned}$$

In the above case, the single edge graph F'' is defined and the F of each new t-pair is its union with F'. Similarly, the function α'' encodes the trivial transfer costs in F''. Also, α is updated so that it includes the fusion of the transfer costs of α and α''.

- If t is an *forget node*. Let v be the forget vertex and let t' be the child of t. Then R_t contains every t-pair (F, α) such that there exists $(F', \alpha') \in \mathcal{R}_{t'}$ where:

  - if t is not the root of Y, then the connected component of F' containing v also contains another element $v' \in X_t$ (this is necessary as X_t should always be good for F),
  - $F = \mathsf{Reduce}(F', X_t)$, with associated reduce function φ,
  - we denote by Z the set of every edge and every vertex that is in F' but not in F. Moreover, if $\varphi(v)$ is a vertex, then we further set $Z \leftarrow Z \cup \{\varphi(v)\}$. Notice also that if $z \in Z$, then $\varphi(z) = \varphi(v)$. Then $\alpha = \alpha'|_{X_t \times (X_t^F \setminus \{\varphi(v)\})} \cup \big\{\big((x, \varphi(v)), \max_{y \in Z} \alpha'(x, y)\big) \mid x \in X_t\big\}$.

  Notice that F is further reduced because v has been "forgotten" in X_t. This may change the status of v as follows: either v is not any more in F or v is still in F but it is not a t-terminal. In the first case $\varphi(v)$ is either a vertex or an edge of F and in the second $\varphi(v) = v$. In any case we should update the values of $\alpha(x, \phi(v))$ for every $x \in X_t$ to the maximum transition cost (with respect to α') from x to some element of Z.

- If t is an *join node*. Let t' and t'' be the children of t. We define

$$\begin{aligned} R_t \quad = \quad & R_{t'} \cup \{(F, \alpha) \mid (F, \alpha) \text{ is admissible, } F \text{ is a forest, and there exist two} \\ & \text{pairs } (F', \alpha') \in R_{t'} \text{ and } (F'', \alpha'') \in R_{t''} \text{ such} \\ & \text{that } F = F' \cup F'' \text{ and } \alpha = \alpha' \oplus_{\beta_t} \alpha''\}. \end{aligned}$$

  The above case is very similar to the case of the edge-introduce node. The only difference is that now F'' is now taken from $\mathcal{R}_{t''}$.

Taking into account Claim 1 on the bound of the size of $\mathcal{R}_t$, it is easy to verify that, in each of the above cases, R_t can be computed in $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$ steps. So we can solve our problem in time $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})} \cdot n$, and the theorem follows. $\square$

6 Polynomially bounded costs

So far, we have completely characterized the parameterized complexity of the Diameter-Tree problem for any combination of the three parameters k, tw, and Δ. In this section we focus on the special case when the maximum cost value is polynomially bounded by n. The following corollary is an immediate consequence of Theorem 5.

Corollary 2. *If the maximum cost value is polynomially bounded by n, the* Diameter-Tree *problem is in* XP *parameterized by* tw *and* Δ.

From Corollary 2, a natural question is whether the Diameter-Tree problem is FPT or $\mathsf{W[1]}$-hard parameterized by tw and Δ, in the case where the maximum cost value is polynomially bounded by n. The next theorem provides an answer to this question.

Theorem 6. *When the maximum cost value is polynomially bounded by n, the* Diameter-Tree *problem is* $\mathsf{W[1]}$*-hard parameterized by* tw *and* Δ.

Proof. We present a parameterized reduction from the Bin Packing problem parameterized by the number of bins. In Bin Packing, we are given n integer item sizes $a_1, \ldots, a_n$ and an integer capacity B, and the objective is to partition the items into a minimum number of bins with capacity B. Jansen *et al.* [26] proved that Bin Packing is $\mathsf{W[1]}$-hard parameterized by the number of bins in the solution, even when all item sizes are bounded by a polynomial in the input size. Equivalently, this version of the problem corresponds to the case where the item sizes are given in *unary* encoding; this is why it is called Unary Bin Packing in [26].

Given an instance $(\{a_1, a_2, \ldots, a_n\}, B, k)$ of Unary Bin Packing, where k is the number of bins in the solution and where we can assume that $k \geqslant 2$, we create an instance (G, χ, c) of Diameter-Tree as follows. The graph G contains a vertex r and, for $i \in [n]$ and $j \in [k]$, we add to G vertices v_i, ℓ_j^i, r_j^i and edges $\{r, \ell_j^1\}$, $\{v_i, \ell_j^i\}$, $\{v_i, r_j^i\}$, and $\{\ell_j^i, r_j^i\}$. Finally, for $i \in [n-1]$ and $j \in [k]$, we add the edge $\{r_j^i, \ell_j^{i+1}\}$. Let G' be the graph constructed so far; see Figure 6 for an illustration.

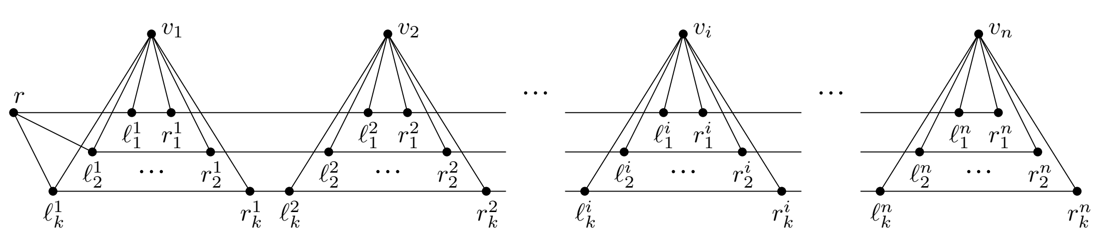

Figure 6: Graph G' built in the reduction of Theorem 6. Reload costs are not depicted.

Similarly to the proof of Theorem 3, we define G to be the graph obtained by taking two disjoint copies of G' and identifying vertex r of both copies. Note that G can be clearly built in polynomial time, and that $\mathsf{tw}(G) \leqslant k + 1$ and $\Delta(G) = 2k$

(since we assume $k \geqslant 2$). Therefore, $\mathsf{tw}(G) + \Delta(G)$ is indeed bounded by a function of k, as required. (Again, the claimed bound on the treewidth can be easily seen by building a *path* decomposition of G with consecutive bags of the form $\{v_i, \ell_1^i, \ell_2^i, \ldots, \ell_k^i, r_1^i\}, \{v_i, \ell_1^i, \ell_2^i, \ldots, \ell_{k-1}^i, r_1^i, r_2^i\}, \{v_i, \ell_1^i, \ell_2^i, \ldots, \ell_{k-2}^i, r_1^i, r_2^i, r_3^i\}, \ldots$)

Let us now define the coloring χ and the cost function c. Once more, for simplicity, we associate a distinct color with each edge of G, and thus it is enough to describe the cost function c for every pair of incident edges of G. The cost function is symmetric for both copies of G', so we just focus on one copy. For $i \in [n]$, let e_1, e_2 be two distinct edges containing vertex v_i. We set $c(e_1, e_2) = 2B+1$ unless $e_1 = \{v_i, \ell_j^i\}$ and $e_2 = \{v_i, r_j^i\}$ for some $j \in [k]$, in which case we set $c(e_1, e_2) = a_i$. The cost associated with any other pair of edges of G is set to 0. Note that, as $(\{a_1, a_2, \ldots, a_n\}, B, k)$ is an instance of Unary Bin Packing, the reload costs of the instance (G, χ, c) of Diameter-Tree are polynomially bounded by $|V(G)|$.

We claim that $(\{a_1, a_2, \ldots, a_n\}, B, k)$ is a Yes-instance of Unary Bin Packing if and only if G has a spanning tree with diameter at most $2B$.

Assume first that $(\{a_1, a_2, \ldots, a_n\}, B, k)$ is a Yes-instance of Unary Bin Packing, and let $S_1, \ldots, S_k$ be the k subsets of $\{1, \ldots, n\}$ defining the k bins in the solution. We define a spanning tree T of G with $\mathsf{diam}(T) \leqslant 2B$ as follows. For each of the two copies of G', tree T contains, for $i \in [n-1]$ and $j \in [k]$, edges $\{r, \ell_j^1\}$ and $\{r_j^i, \ell_j^{i+1}\}$. For $i \in [n-1]$, if the item a_i belongs to the set S_j, we add to T the two edges $\{v_i, \ell_j^i\}$ and $\{v_i, r_j^i\}$; otherwise we add to T the edge $\{\ell_j^i, r_j^i\}$. Since the total item size of each bin in the solution of Unary Bin Packing is at most B, it can be easily checked that T is a spanning tree of G with $\mathsf{diam}(T) \leqslant 2B$.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leqslant 2B$, and we proceed to define a solution $S_1, \ldots, S_k$ of Unary Bin Packing. Let T_1 and T_2 be the restriction of T to the two copies of G'. By the choice of the reload costs and since $\mathsf{diam}(T) \leqslant 2B$, for every $i \in [n]$ and every $x \in \{1, 2\}$, tree T_x contains the two edges $\{v_i, \ell_j^i\}$ and $\{v_i, r_j^i\}$ for some $j \in [k]$, and none of the other edges incident with vertex v_i. Therefore, for every $x \in \{1, 2\}$, tree T_x consists of k paths sharing vertex r. This implies that $\mathsf{diam}(T) \geqslant \frac{1}{2}\mathsf{diam}(T_1) + \frac{1}{2}\mathsf{diam}(T_2)$, and since $\mathsf{diam}(T) \leqslant 2B$, it follows that there exists $x \in \{1, 2\}$ such that $\mathsf{diam}(T_x) \leqslant B$. Assume without loss of generality that $x = 1$, i.e., that $\mathsf{diam}(T_1) \leqslant B$. We define the bins $S_1, \ldots, S_k$ as follows. For every $i \in [n]$, if T_1 contains the two edges $\{v_i, \ell_j^i\}$ and $\{v_i, r_j^i\}$, we add item a_i to the bin S_j. Let us verify that this defines a solution of Unary Bin Packing. Indeed, assume for contradiction that for some $j \in [k]$, the total item size in bin S_j exceeds B. As bin S_j corresponds to one of the k paths in tree T_1, the diameter of this path would also exceed B, contradicting the fact that $\mathsf{diam}(T_1) \leqslant B$. The theorem follows. $\square$

7 Concluding remarks

We provided an accurate picture of the (parameterized) complexity of the Diameter-Tree problem for any combination of the parameters k, tw, and Δ, distinguishing whether the reload costs are polynomial or not. Some questions still remain open. First of all, in the hardness result of Theorem 3, we already mentioned that the bound $\Delta \leqslant 3$ is tight, but the bound $\mathsf{tw} \leqslant 3$ might be improved to $\mathsf{tw} \leqslant 2$. A relevant question is whether the problem admits polynomial kernels parameterized by $k + \mathsf{tw} + \Delta$ (recall that it is FPT by Theorem 5). Theorem 6 motivates the following question: when all reload costs are bounded by a *constant*, is the Diameter-Tree problem FPT parameterized by $\mathsf{tw} + \Delta$? It also makes sense to consider the *color-degree* as a parameter (cf. [23]). Finally, we may consider other relevant width parameters, such as pathwidth (note that the hardness results of Theorems 1, 3, and 6 also hold for pathwidth), cliquewidth, treedepth, or tree-cutwidth.

Acknowledgement. We would like to thank the anonymous reviewers for helpful comments that improved the presentation of the manuscript.

References

[1] S. Agarwal and S. De, *Dynamic spectrum access for energy-constrained CR: Single channel versus switched multichannel*, IET Commun. **10** (2016), 761–769.

[2] E. Amaldi, G. Galbiati, and F. Maffioli, *On minimum reload cost paths, tours, and flows*, Networks **57** (2011), 254–260.

[3] S. Arkoulis, E. Anifantis, V. Karyotis, S. Papavassiliou, and N. Mitrou, *On the optimal, fair and channel-aware cognitive radio network reconfiguration*, Comput. Networks **57** (2013), 1739–1757.

[4] S. Bayhan and F. Alagoz, *Scheduling in centralized cognitive radio networks for energy efficiency*, IEEE Trans. Vehicular Technology **62** (2013), 582–595.

[5] S. Bayhan, S. Eryigit, F. Alagoz, and T. Tugcu, *Low complexity uplink schedulers for energy-efficient cognitive radio networks*, IEEE Wireless Commun. Lett. **2** (2013), 363–366.

[6] A.P. Bianzino, C. Chaudet, D. Rossi, and J.L. Rougier, *A survey of green networking research*, IEEE Commun. Surveys Tutorials **14** (2012), 3–20.

[7] H.L. Bodlaender, P.G. Drange, M.S. Dregi, F.V. Fomin, D. Lokshtanov, and M. Pilipczuk, *A $c^k n$ 5-approximation algorithm for treewidth*, SIAM J. Comput. **45** (2016), 317–378.

[8] A. Celik and A.E. Kamal, *Green cooperative spectrum sensing and scheduling in heterogeneous cognitive radio networks*, IEEE Trans. Cognitive Commun. Networking **2** (2016), 238–248.

[9] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015.

[10] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J.M.M. van Rooij, and J.O. Wojtaszczyk, *Solving connectivity problems parameterized by treewidth in single exponential time*, Proc. of the 52nd Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2011, pp. 150–159.

[11] C. Desset, N. Ahmed, and A. Dejonghe, *Energy savings for wireless terminals through smart vertical handover*, Proc. of IEEE International Conference on Communications, 2009, pp. 1–5.

[12] R. Diestel, *Graph Theory* Vol. 173, Springer-Verlag, 4th edition, 2010.

[13] R.G. Downey and M.R. Fellows, *Fundamentals of Parameterized Complexity*, Texts in Computer Science, Springer, 2013.

[14] S. Eryigit, S. Bayhan, and T. Tugcu, *Channel switching cost aware and energy-efficient cooperative sensing scheduling for cognitive radio networks*, Proc. of IEEE International Conference on Communications (ICC), 2013, pp. 2633–2638.

[15] G. Galbiati, *The complexity of a minimum reload cost diameter problem*, Discr. Appl. Math. **156** (2008), 3494–3497.

[16] G. Galbiati, S. Gualandi, and F. Maffioli, *On minimum changeover cost arborescences*, Proc. of the 10th International Symposium on Experimental Algorithms (SEA), Vol. 6630 of *LNCS*, 2011, pp. 112–123.

[17] G. Galbiati, S. Gualandi, and F. Maffioli, *On minimum reload cost cycle cover*, Discr. Appl. Math. **164** (2014), 112–120.

[18] I. Gamvros, L. Gouveia, and S. Raghavan, *Reload cost trees and network design*, Networks **59** (2012), 365–379.

[19] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[20] L. Gourvès, A. Lyra, C. Martinhon, and J. Monnot, *The minimum reload s-t path, trail and walk problems*, Discr. Appl. Math. **158** (2010), 1404–1417.

[21] D. Gözüpek, S. Buhari, and F. Alagöz, *A spectrum switching delay-aware scheduling algorithm for centralized cognitive radio networks*, IEEE Trans. Mobile Comput. **12** (2013), 1270–1280.

[22] D. Gözüpek, S. Özkan, C. Paul, I. Sau, and M. Shalom, *Parameterized complexity of the MINCCA problem on graphs of bounded decomposability*, Theor. Comput. Sci. **690** (2017), 91–103.

[23] D. Gözüpek, H. Shachnai, M. Shalom, and S. Zaks, *Constructing minimum changeover cost arborescenses in bounded treewidth graphs*, Theor. Comput. Sci. **621** (2016), 22–36.

[24] D. Gözüpek and M. Shalom, *Complexity of edge coloring with minimum reload/changeover costs*, Networks **73** (2019), 344–357.

[25] D. Gözüpek, M. Shalom, A. Voloshin, and S. Zaks, *On the complexity of constructing minimum changeover cost arborescences*, Theor. Comput. Sci. **540** (2014), 40–52.

[26] K. Jansen, S. Kratsch, D. Marx, and I. Schlotter, *Bin packing with fixed number of bins revisited*, J. Comput. System Sci. **79** (2013), 39–49.

[27] F.V. Jensen, *Bayesian Networks and Decision Graphs*, Springer, 2001.

[28] T. Kloks, *Treewidth, Computations and Approximations* Vol. 842 of *LNCS*, Springer, 1994.

[29] V.R. Konda and T.Y. Chow, *Algorithm for traffic grooming in optical networks to minimize the number of transceivers*, Proc. of IEEE Workshop on High Performance Switching and Routing, 2001, pp. 218–221.

[30] S.J. Lauritzen and D.J. Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems*, J. Royal Stat. Society. Ser. B (Methodological) **50** (1988), 157–224.

[31] N. Shami and M. Rasti, *A joint multi-channel assignment and power control scheme for energy efficiency in cognitive radio networks*, Proc. of IEEE Wireless Communications and Networking Conference (WCNC), 2016, pp. 1–6.

[32] C.A. Tovey, *A simplified NP-complete satisfiability problem*, Discr. Appl. Math. **8** (1984), 85–89.

[33] H.C. Wirth and J. Steffan, *Reload cost problems: Minimum diameter spanning tree*, Discr. Appl. Math. **113** (2001), 73–85.