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

Julien Baste[1], Didem Gözüpek[2], Christophe Paul[3], Ignasi Sau[3,4], Mordechai Shalom[5,6], and Dimitrios M. Thilikos[3,7]

1 **Université de Montpellier, LIRMM, Montpellier, France**
2 **Department of Computer Engineering, Gebze Technical University, Kocaeli, Turkey**
3 **AlGCo project team, CNRS, LIRMM, France**
4 **Departamento de Matemática, Universidade Federal do Ceará, Fortaleza, Brazil**
5 **TelHai College, Upper Galilee, 12210, Israel**
6 **Department of Industrial Engineering, Boğaziçi University, Istanbul, Turkey**
7 **Department of Mathematics, National and Kapodistrian University of Athens, Greece**

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.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, C.2.1 Network Architecture and Design, G.2.2 Graph Theory.

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

Digital Object Identifier 10.4230/LIPIcs...

* Emails of authors: `baste@lirmm.fr`, `didem.gozupek@gtu.edu.tr`, `paul@lirmm.fr`, `sau@lirmm.fr`, `cmshalom@telhai.ac.il`, `sedthilk@thilikos.info` .
This work has been supported by the bilateral research program of CNRS and TUBITAK under grant no.114E731, PASTA project of Université de Montpellier, TUBITAK 2221 programme, and by project DEMOGRAPH (ANR-16-CE40-0028).

1 Introduction

Numerous network optimization problems can be modeled by edge-colored graphs. Wirth and Steffan introduced in [31] 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 requires 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, 13, 20, 29]) 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 increment. 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].

Reload cost concept finds applications also 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 [14]. 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 [14].

Recent works in the literature focused on numerous problems related to the reload cost concept: the minimum reload cost cycle cover problem [16], the problems of finding a path, trail or walk with minimum total reload cost between two given vertices [19], the problem of finding a spanning tree that minimizes the sum of reload costs of all paths between all pairs of vertices [17], various path, tour, and flow problems related to reload costs [2], the minimum changeover cost arborescence problem [15, 21, 22, 24], and problems related to finding a proper edge coloring of the graph so that the total reload cost is minimized [23].

The work in [31], 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 an 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 availabilities 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 [31] 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 [14] 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 combinations 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 [26]. 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 [28]. 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
	$k+\mathsf{tw}$	$k+\Delta$	$\mathsf{tw}+\Delta$	$k+\mathsf{tw}+\Delta$	cases
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 4)
DIAMETER-TREE with poly costs	✓	✓	XP (Thm 5) W[1]-hard (Thm 7)	✓	✓

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 outer-planar 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 [31]. 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 $\Delta \leq 3$, 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 $\Delta \leq 3$ in the general case where the triangle inequality is not necessarily satisfied, which had been left open in previous work [14, 31]. Note that $\Delta \leq 3$ is best possible, as DIAMETER-TREE can be easily solved on graphs with $\Delta \leq 2$.

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 $\mathsf{tw} \leq 3$ and $\Delta \leq 3$.

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 [18], 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.* [25].

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

Further research. In the hardness result of Theorem 3, the bound $\Delta \leq 3$ is tight, but the bound $\mathsf{tw} \leq 3$ might be improved to $\mathsf{tw} \leq 2$. 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 7 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. [22]). Finally, we may consider other relevant width parameters, such as pathwidth (note that the hardness results of Theorems 1, 3, and 7 also hold for pathwidth), cliquewidth, treedepth, or tree-cutwidth.

Organization of the paper. We start in Section 2 with some preliminaries about the DIAMETER-TREE problem. Basic definitions about graphs, parameterized complexity, and tree decompositions can be found in Appendix A. In Section 3 we provide the para-NP-hardness results, and in Section 4 we present the polynomial-time algorithm on cactus graphs and the FPT algorithm on general graphs parameterized by $k + \mathsf{tw} + \Delta$. In Section 5 we focus on the case where the reload costs are polynomially bounded. Due to lack of space, the proof of the results marked with '[$\star$]' have been moved to the appendix.

2 Reload costs and definition of the problem

For reload costs, we follow the notation and terminology defined by [31]. 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 for simplicity to be symmetric. The cost of traversing two incident edges e_1, e_2 is $c(e_1, e_2) := c(\chi(e_1), \chi(e_2))$. By definition, reload costs at the endpoints of a path equal zero. Consequently, the reload cost of a path with one edge also equals zero. The *reload cost* of a path P of length $\ell \geq 2$ with edges $e_1, e_2, \ldots, e_\ell$ is defined as $c(P) := \sum_{i=2}^{\ell} c(e_{i-1}, e_i)$. The induced reload cost distance function is given by $\mathsf{dist}_G^c(u, v) = \min\{c(P) \mid P \text{ is a path from } u \text{ to } v \text{ in } G\}$. The *diameter* of a tree T is $\mathsf{diam}(T) := \max_{u,v \in V} \mathsf{dist}_T^c(u, v)$, where for notational convenience we assume that the edge-coloring function χ and the reload cost function c are clear from the context.

The problem we study in this paper can be formally 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 $c(e_1, e_3) \leq c(e_1, e_2) + c(e_2, e_3)$, we say that the reload cost function c satisfies the *triangle inequality*. This assumption is sometimes used in practical applications [31].

Throughout the paper, we let n, Δ, and tw denote 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 let also k denote the desired cost of a solution.

3 Para-NP-hardness results

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

▶ **Theorem 1.** *The Diameter-Tree problem is NP-hard on outerplanar graphs with only one vertex of degree greater 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 parameterized by tw and k.*

Proof. We present a simple 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 there is no clause in φ that contains a literal and its negation. The graph G contains a distinguished vertex r and, for each clause $c_j = (\ell_1 \vee \ell_2 \vee \ell_3)$, we add a clause gadget C_j consisting of three vertices $v_{\ell_1}^j, v_{\ell_2}^j, v_{\ell_3}^j$ and five edges $\{r, v_{\ell_1}^j\}$, $\{r, v_{\ell_2}^j\}$, $\{r, v_{\ell_3}^j\}$, $\{v_{\ell_1}^j, v_{\ell_2}^j\}$, and $\{v_{\ell_2}^j, v_{\ell_3}^j\}$. 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; see Figure 1 for an illustration.

Let us now define 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

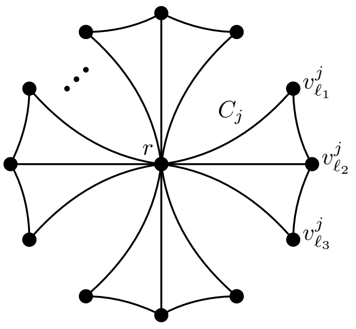

Figure 1 Example of the graph G built in the reduction of Theorem 1.

cost functions. We will use just three different costs: 1, 5 and 10. 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.

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 associated with a path from r to a leaf of this tree is always 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, fix a satisfying assignment ψ of φ, and let us construct a spanning tree T of G with diameter at most 9. For each clause c_j, the tree T^j is the tree spanning C_j and containing the edge between r and an arbitrarily chosen 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, it holds that $\mathsf{diam}(T) \leq 9$.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leq 9$. Then, the reload cost incurred at r by any path traversing it is at most 5 since otherwise $\mathsf{diam}(T) \geq 10$. 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. ◀

We proceed with the para-NP-hardness result with parameter $k + \Delta$.

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

Proof. We present a reduction from the restriction of 3-Sat to formulas where each variable occurs in at most three clauses; this problem was proved to be NP-complete by Tovey [30]. It

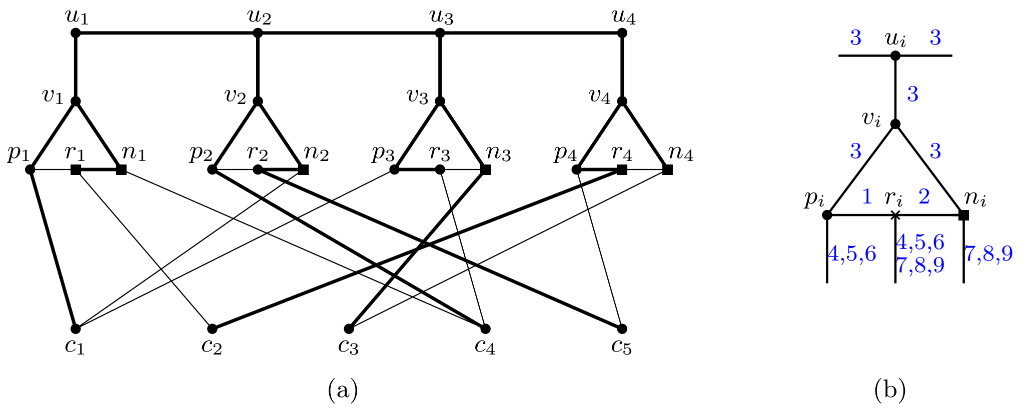

Figure 2 (a) Graph G described in the reduction of Theorem 2 for the formula $\varphi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee x_4)$. The vertices p_i, r_i, n_i 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 edge edge of G are depicted in blue.

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 [30].

We may assume that each variable occurs at least once positively and at least once negatively, as otherwise we may set such variable x to the value that satisfies all clauses in which it appears, and delete x together with those clauses from the formula. We may also assume that each variable occurs *exactly* three times in the given formula φ. Indeed, let x be a variable occurring exactly two times in the formula. We create a new variable y and we add to φ two clauses $(x \vee y)$ and $(y \vee \overline{y})$. Let φ' be the new formula. Clearly φ and φ' are equivalent, and both x and y occur three times in φ'. Applying these operations exhaustively clearly results in an equivalent formula where each variable occurs exactly three times. Summarizing, we may assume the following property:

✠ *Each variable occurs exactly three times in the given formula φ of* 3-Sat. *Moreover, 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 $\Delta(G) \leq 3$ as follows. Let the variables in φ be $x_1, \ldots, x_n$. For every $i \in [n]$, we add to G a *variable gadget* consisting of five vertices u_i, v_i, p_i, r_i, n_i and five edges $\{u_i, v_i\}, \{v_i, p_i\}, \{p_i, r_i\}, \{r_i, n_i\}$, and $\{n_i, v_i\}$. For every $i \in [n-1]$, we add the edge $\{u_i, u_{i+1}\}$. For every $j \in [m]$, the clause gadget in G consists of a single vertex c_j. We now proceed to explain how we connect the variable and the clause gadgets. For each variable x_i, we connect vertex p_i (resp. n_i) to one of the vertices corresponding to a clause of φ in which x_i appears positively (resp. negatively). Finally, we connect vertex r_i to the remaining clause in which x_i appears (positively or negatively). Note that these connections are well-defined because of property ✠. This completes the construction of G, and note that it indeed holds that $\Delta(G) \leq 3$; see Figure 2(a) for an example of the construction of G for a specific satisfiable formula φ with $n = 4$ and $m = 5$.

Let us now define the coloring χ and the cost function c. We use nine colors $1, 2, \ldots, 9$ associated with the edges of G as follows. For $i \in [n]$, we set $\chi(\{p_i, r_i\}) = 1$ and $\chi(\{r_i, n_i\}) = 2$, and all edges incident to u_i or v_i have color 3. Finally, for $j \in [m]$, we color the edges containing c_j with colors in $\{4, 5, 6, 7, 8, 9\}$, so that incident edges get different colors, and edges corresponding to positive (resp. negative) occurrences get colors in $\{4, 5, 6\}$ (resp.

$\{7,8,9\}$); note that such a coloring always exists as each clause contains at most three variables; see Figure 2(b). We will use only two costs, namely 0 and 1, and recall that we consider just symmetric cost functions. We set $c(1,2)=1$, $c(1,i)=1$ for every $i \in \{4,5,6\}$, $c(2,i)=1$ for every $i \in \{7,8,9\}$, and $c(i,j)=1$ for every distinct $4 \leq i,j \leq 9$. All other costs are set to 0. The following claim concludes the proof.

▶ Claim 1. [⋆] φ is satisfiable if and only if G contains a spanning tree with diameter 0. ◀

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 [31].

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

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

Proof. We present a reduction from the Partition problem, which is a typical example of a *weakly* NP-complete problem [18]. 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 $\{u'_{i-1}, d'_{i-1}, u_i, d_i\}, \{u_i, d_i, m_i, u'_i\}, \{d_i, m_i, u'_i, m'_i\}, \{d_i, u'_i, m'_i, d'_i\}, \ldots$.)

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 $1 \leq i \leq n-1$, 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 $1 \leq i \leq n$, 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 3 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. The following claim concludes the proof.

▶ Claim 2. [⋆] The instance S of Partition is a Yes-instance if and only if G has a spanning tree with diameter at most B. ◀

4 Polynomial and FPT algorithms

We start this section by presenting the polynomial-time algorithm to solve the Diameter-Tree problem on cactus graphs, equivalently called *cacti*. We first need some definitions.

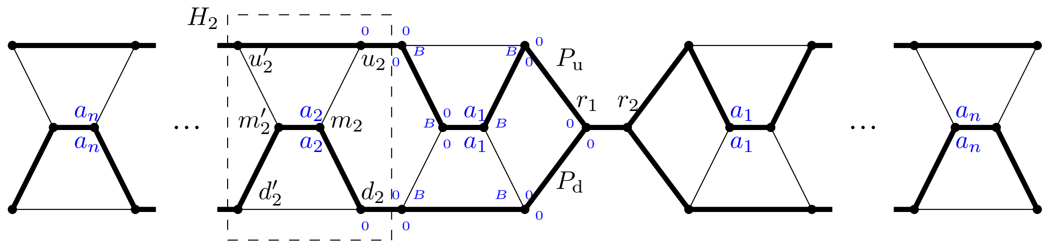

Figure 3 Graph G built in the reduction of Theorem 3, where the reload costs are depicted in blue at the angle between the two corresponding edges. For better visibility, not all costs and vertex labels are depicted. The typical shape of a solution spanning tree is highlighted with thicker edges.

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 T whose nodes are the cut vertices and the blocks of G. Every cut vertex is adjacent in T to all the blocks that contain it. Two blocks share at most one vertex. 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 as *cycle block* and *edge block*, respectively. It is well-known that cacti have treewidth at most 2. Given a forest F and two vertices x and y, we define $\texttt{cost}_F(x,y)$ to be $\mathsf{dist}^c_T(x,y)$ if x and y are in the same tree T of F and where c is the given reload cost function, and $\perp$ otherwise. Given a tree T and a vertex $v \in V(T)$, we define the *eccentricity* of v in T to be $\max_{v' \in V(T)} \texttt{cost}_T(v,v')$.

We present a polynomial-time algorithm that solves the decision version of the problem, which we call DIAMETER-TREE*: the input is an edge-colored graph G and an integer k, and the objective is to decide whether the input graph G has a spanning tree with reload cost diameter at most k. The algorithm to solve DIAMETER-TREE* uses dynamic programming on the block tree of the input graph.

As we aim at a *truly* polynomial-time algorithm to solve DIAMETER-TREE, we *cannot* afford to solve the decision version for all values of k. To overcome this problem, we perform a double *binary search* on the possible solution values and two appropriate eccentricities, resulting (skipping many technical details) in an extra factor of $(\log \texttt{opt})^2$ in the running time of the algorithm, where $\texttt{opt}$ is the diameter of a minimum cost spanning tree. This yields a polynomial-time algorithm solving DIAMETER-TREE in cactus graphs.

Roughly speaking, the algorithm first fixes an arbitrary non-cut vertex r of G and the block B_r that contains it. Then it 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. The intuition behind the instances of 2-SAT created by the algorithm is the following.

Suppose that we are dealing with a cycle block B of the block tree of G (the case of an edge block being easier). Note that any spanning tree of G contains all edges of B except one. Let G_B be the graph processed so far (including B). For each potential partial solution $\mathcal{Q}$ in G_B, we associate, with each edge e of B, a variable that indicates that e is the non-picked edge by the solution in B. Now, for any *two* such variables corresponding to intersecting blocks, we add to the formula of 2-SAT essentially two types of clauses: the first set of clauses, namely ϕ_1, guarantees that the non-picked edges (corresponding to the variables set to $\texttt{true}$ in the eventual assignment) indeed define a spanning tree of G_B, while the second one, namely ϕ_2, forces this solution to have diameter and eccentricity not exceeding the given budget k. The fact the G is a cactus allows to prove that these constraints containing only two variables are enough to compute an optimal solution in G_B. Full details can be found in the appendix.

▶ **Theorem 4.** [⋆] *The* Diameter-Tree *problem can be solved in polynomial time on cacti.*

In the following theorem 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.

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

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

From Corollary 6, a natural question is whether the Diameter-Tree problem is FPT or 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 7.** *When the maximum cost value is polynomially bounded by n, the* Diameter-Tree *problem is* 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.* [25] proved that Bin Packing is W[1]-hard parameterized by the number of bins in the solution, even when all item sizes are bounded by a polynomial of 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 [25].

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 $k \geq 2$, 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^i_j, r^i_j and edges $\{r, \ell^1_j\}$, $\{v_i, \ell^i_j\}$, $\{v_i, r^i_j\}$, and $\{\ell^i_j, r^i_j\}$. Finally, for $i \in [n-1]$ and $j \in [k]$, we add the edge $\{r^i_j, \ell^{i+1}_j\}$. Let G' be the graph constructed so far; see Figure 4 for an illustration.

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 $\mathsf{tw}(G) \leq k+1$ and $\Delta(G) = 2k$ (since we assume $k \geq 2$). 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 $\{v_i, \ell^i_1, \ell^i_2, \ldots, \ell^i_k, r^i_1\}, \{v_i, \ell^i_1, \ell^i_2, \ldots, \ell^i_{k-1}, r^i_1, r^i_2\}, \{v_i, \ell^i_1, \ell^i_2, \ldots, \ell^i_{k-2}, r^i_1, r^i_2, r^i_3\}, \ldots$.)

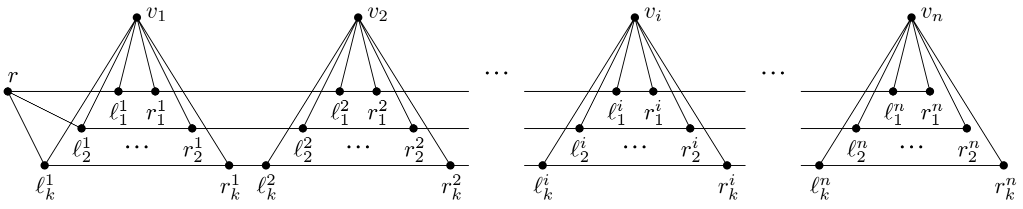

Figure 4 Graph G' built in the reduction of Theorem 7. The reload costs are not depicted.

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 $|V(G)|$. Again, the following claim concludes the proof.

▶ Claim 3. [$\star$] $(\{a_1, a_2, \ldots, a_n\}, B, k)$ is a YES-instance of UNARY BIN PACKING if and only if G has a spanning tree with diameter at most $2B$. ◀

References

1 S. Agarwal and S. De. Dynamic spectrum access for energy-constrained cr: single channel versus switched multichannel. *IET Communications*, 10(7):761–769, 2016.

2 E. Amaldi, G. Galbiati, and F. Maffioli. On minimum reload cost paths, tours, and flows. *Networks*, 57(3):254–260, 2011.

3 S. Arkoulis, E. Anifantis, V. Karyotis, S. Papavassiliou, and N. Mitrou. On the optimal, fair and channel-aware cognitive radio network reconfiguration. *Computer Networks*, 57(8):1739–1757, 2013.

4 S. Bayhan and F. Alagoz. Scheduling in centralized cognitive radio networks for energy efficiency. *IEEE Transactions on Vehicular Technology*, 62(2):582–595, 2013.

5 S. Bayhan, S. Eryigit, F. Alagoz, and T. Tugcu. Low complexity uplink schedulers for energy-efficient cognitive radio networks. *IEEE Wireless Communications Letters*, 2(3):363–366, 2013.

6 A. P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier. A survey of green networking research. *IEEE Communications Surveys & Tutorials*, 14(1):3–20, 2012.

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 Journal on Computing*, 45(2):317–378, 2016.

8 A. Celik and A. E. Kamal. Green cooperative spectrum sensing and scheduling in heterogeneous cognitive radio networks. *IEEE Transactions on Cognitive Communications and Networking*, 2(3):238–248, 2016.

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. In *Proc. of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159. IEEE Computer Society, 2011.

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

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

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

14 G. Galbiati. The complexity of a minimum reload cost diameter problem. *Discrete Applied Mathematics*, 156(18):3494–3497, 2008.

15 G. Galbiati, S. Gualandi, and F. Maffioli. On minimum changeover cost arborescences. In *Proc. of the 10th International Symposium on Experimental Algorithms (SEA)*, volume 6630 of *LNCS*, pages 112–123, 2011.

16 G. Galbiati, S. Gualandi, and F. Maffioli. On minimum reload cost cycle cover. *Discrete Applied Mathematics*, 164:112–120, 2014.

17 I. Gamvros, L. Gouveia, and S. Raghavan. Reload cost trees and network design. *Networks*, 59(4):365–379, 2012.

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

19 L. Gourvès, A. Lyra, C. Martinhon, and J. Monnot. The minimum reload s-t path, trail and walk problems. *Discrete Applied Mathematics*, 158(13):1404–1417, 2010.

20 D. Gözüpek, S. Buhari, and F. Alagöz. A spectrum switching delay-aware scheduling algorithm for centralized cognitive radio networks. *IEEE Transactions on Mobile Computing*, 12(7):1270–1280, 2013.

21 D. Gözüpek, S. Özkan, C. Paul, I. Sau, and M. Shalom. Parameterized complexity of the MINCCA problem on graphs of bounded decomposability. In *Proc. of the 42nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 9941 of *LNCS*, pages 195–206, 2016. Full version available at `arXiv:1509.04880`.

22 D. Gözüpek, H. Shachnai, M. Shalom, and S. Zaks. Constructing minimum changeover cost arborescenses in bounded treewidth graphs. *Theoretical Computer Science*, 621:22–36, 2016.

23 D. Gözüpek and M. Shalom. Edge coloring with minimum reload/changeover costs. *Preprint available at* `arXiv:1607.06751`, 2016.

24 D. Gözüpek, M. Shalom, A. Voloshin, and S. Zaks. On the complexity of constructing minimum changeover cost arborescences. *Theoretical Computer Science*, 540:40–52, 2014.

25 K. Jansen, S. Kratsch, D. Marx, and I. Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013.

26 F. V. Jensen. *Bayesian Networks and Decision Graphs.* Springer, 2001.

27 T. Kloks. *Treewidth. Computations and Approximations.* Springer-Verlag LNCS, 1994.

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

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

30 C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89, 1984.

31 H.-C. Wirth and J. Steffan. Reload cost problems: minimum diameter spanning tree. *Discrete Applied Mathematics*, 113(1):73–85, 2001.

A Preliminaries about graphs, parameterized complexity, and tree decompositions

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. We also define $N_G[S] = S \cup \{x \mid \exists y \in S : \{x, y\} \in E(G)\}$. For a graph G and an edge $e \in E(G)$, we let $G - e = (V(G), E(G) \setminus e)$. Given a graph G and a set $S \subseteq V(G)$, we say that S is *good for* G if each connected component of G contains at least one vertex of S. Given two integers i and j with $i \leq j$, we denote by $[i, j]$ the set of all integers k such that $i \leq k \leq j$. For an integer $i \geq 1$, we denote by $[i]$ the set of all integers k such that $1 \leq k \leq i$.

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

A parameterized problem is *fixed-parameter tractable* (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an FPT *algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^c$. For instance, the Vertex Cover problem parameterized by the size of the solution is FPT.

A parameterized problem is in XP if there exists an algorithm $\mathcal{A}$ and two computable functions f and g such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an XP *algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^{g(k)}$. For instance, the Clique problem parameterized by the size of the solution is in XP.

A parameterized problem with instances of the form $I = (x, k)$ is para-NP-*hard* if it is NP-hard for some fixed *constant* value of the parameter k. For instance, the Vertex Coloring problem parameterized by the number of colors is para-NP-hard. Note that, unless P = NP, a para-NP-hard problem cannot be in XP, hence it cannot be FPT either.

Within parameterized problems, the class W[1] may be seen as the parameterized equivalent to the class NP of classical optimization problems. Without entering into details (see [9] for the formal definitions), a parameterized problem being W[1]-*hard* can be seen as a strong evidence that this problem is *not* FPT. For instance, the Clique problem parameterized by the size of the solution is the canonical example of a W[1]-hard problem. To transfer 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 in time $f(k) \cdot |I|^c$, for some computable function f and a function c, an equivalent instance $I' = (x', k')$ of the target problem, such that 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 at most w.

B Proof of Claim 2

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 3). 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 $\mathsf{diam}(T) \leq B$. 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 $\mathsf{diam}(T) \leq B - 1$, it can be verified that, for $j \in \{1, 2\}$, $T \cap G_j$ consists of two paths $P^j_\mathrm{u}, P^j_\mathrm{d}$ intersecting at the root r_i. Furthermore, P^j_u (resp. P^j_d) 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^j_u (resp. P^j_d) 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 $d^j_x := \mathsf{diam}(P^j_x) = \sum_{i \in I^j_x} a_i$, where I^j_x is the set of indices $i \in \{1, \ldots, n\}$ such that the edge $\{m_i, m'_i\}$ belongs to path P^j_x. Note also that, for $j \in \{1, 2\}$, by construction we have that $d^j_\mathrm{u} + d^j_\mathrm{d} = \sum_{i=1}^n a_i$, implying in particular that $\max\{d^j_\mathrm{u}, d^j_\mathrm{d}\} \geq \frac{B}{2}$. On the other hand, by the structure of T it holds that

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

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

C Proof of Claim 2

Assume first that φ is satisfiable, fix a satisfying assignment ψ of φ, and let us construct a spanning tree T of G with diameter 0. For every $i \in [n]$, tree T contains all the edges containing vertex u_i or v_i. If variable x_i is set to true by ψ, we include the edge $\{r_i, n_i\}$ to T, and otherwise, that is, if x_i is set to false by ψ, we include the edge $\{p_i, r_i\}$. Finally, for $j \in [m]$, we add to T one of the edges containing c_j that corresponds to a literal satisfying that clause. It can be easily checked that T is a spanning tree of G with diameter 0; see Figure 2(a) for an example.

Conversely, let T be a spanning tree of G with diameter 0. Since the cost associated with any two distinct colors in $\{4, 5, 6, 7, 8, 9\}$ is 1, it follows that, for $j \in [m]$, vertex c_j has degree one in T. Therefore, the variable gadgets need to be connected in T via the vertices u_i, implying that all edges containing u_i, for $i \in [n]$, belong to T. For $i \in [n]$, in order for T to contain all four vertices v_i, p_i, r_i, n_i, by construction of G and since all clause vertices have degree one in T, tree T necessarily contains exactly three out of the four edges of the 4-cycle defined by v_i, p_i, r_i, n_i. Since $c(1, 2) = 1$ and $\mathsf{diam}(T) = 0$, the missing edge is necessarily 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: for $i \in [n]$, if the edge $\{r_i, n_i\}$ belongs to T, we set x_i to true; otherwise, we set x_i to false. We claim that ψ satisfies φ. Indeed, let c_j be a vertex in G corresponding to an arbitrary clause of φ. Since c_j has degree one in T, it is attached to exactly one of the vertices p_i, r_i, n_i for some $i \in [n]$. Suppose that the edge containing c_j corresponds to a positive occurrence of x_i, the other case being symmetric. Then, by construction, necessarily the edge containing c_j is either $\{c_j, p_i\}$ or $\{c_j, r_i\}$. 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 clause corresponding to c_j is satisfied by variable x_i.

D Proof of Theorem 4

We start with a few more definitions needed in the algorithm. Given a graph G, we denote by $\mathcal{S}(G)$ the set of spanning trees of G, and by $\mathcal{B}(G)$ the set of blocks of G. We omit G from the notation if no ambiguity arises. We assume without loss of generality that G is connected. For a block B, we denote by $\mathcal{C}(B)$ the set of blocks that are immediate descendants of B in the block tree. With a slight abuse (since we ignore the cut vertices in the block tree), we will refer to them as the *children* of B. The *parent* of a block B is the first block after B on the path from B to B_r in the block tree. We denote by G_B the subgraph of G induced by the union of all descendants B (including B itself). The *anchor* $a(B)$ of a block B is the cut vertex separating B from its parent if $B \neq B_r$, and r if $B = B_r$.

Let B be a cycle block, $e = \{x, y\}$, and assume, without loss of generality, that $y \neq a(B)$. Clearly, the graph $G_B - e$ is connected. Moreover, $a(B)$ is a cut vertex of $G_B - e$ unless $x = a(B)$. For $z \in \{x, y\}$ we define $S_B^{z,e}$ as the set of vertices that are reachable from z in $G_B - e$ without traversing $a(B)$. See Figure 5 for an illustration. We denote the subgraph of $G_B - e$ induced by $S_B^{z,e}$, as $G_B^{z,e}$. Note that z and $a(B)$ are in $S_B^{z,e}$ and if $x = a(B)$ then $S_B^{x,e} = \{a(B)\}$. Since the degree of $a(B)$ in $G_B - e$ is at most two, a spanning tree T of $G_B - e$ is a union of two spanning trees, a tree $T[S_B^{x,e}]$ spanning $G_B^{x,e}$ and a tree $T[S_B^{y,e}]$ spanning $G_B^{y,e}$. Moreover, $T[S_B^{x,e}]$ and $T[S_B^{y,e}]$ intersect only at $a(B)$.

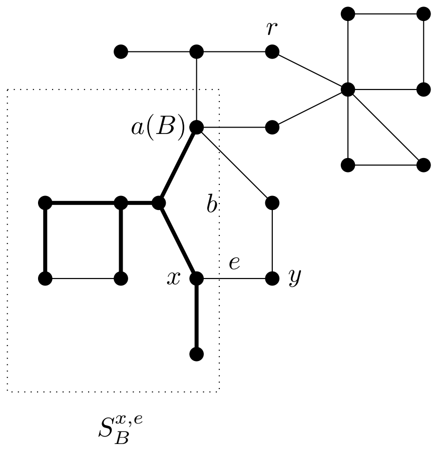

Figure 5 A cactus with 8 blocks, 5 cycle blocks, and 3 edge blocks. The vertices inside the dotted rectangle are the vertices of $S_B^{x,e}$ and the bold path corresponds to a possible $R_B^{x,e}$.

We proceed with the description of the algorithm. At every block B, we compute a function $\lambda_B : E(B) \to \mathcal{S}(G_B) \cup \{\perp\}$ of partial solutions. If B is an edge block consisting of the edge e, then $\lambda_B(e)$ is:

- a spanning tree of G_B,
- of diameter at most k
- that minimizes the eccentricity of $a(B)$,

if such a tree exists, and $\perp$ otherwise.

If B is a cycle block and $e = \{x, y\}$ an edge of B, then $\lambda_B(e)$ is:

- a spanning tree T of $G_B - e$,
- of diameter at most k
- that minimizes the eccentricities of $a(B)$ in both $T[S_B^{x,e}]$ and $T[S_B^{y,e}]$

if such a tree exists, and $\perp$ otherwise. Note that, as $G_B^{x,e}$ and $G_B^{y,e}$ have only the vertex $a(B)$ in common, minimizing the eccentricities of $a(B)$ in $T[S_B^{x,e}]$ and minimizing the eccentricities of $a(B)$ in $T[S_B^{y,e}]$ are two independent objectives.

If for some block B we have $\lambda_B(e) = \perp$ for every edge e of B, then G_B (and therefore G as well) does not contain a spanning tree of diameter at most k. In this case the algorithm stops and returns No. Otherwise, the processing continues until finally B_r is processed successfully and the algorithm returns Yes, since there exists $e \in E(B_r)$ such that $\lambda_{B_r}(e) \neq \perp$ which constitutes a spanning tree of G with diameter at most k.

Given a cycle block B, an edge $e = \{x, y\}$ of B, a subgraph T of G_B, and two integers i and j, we say that T satisfies the (e, i, j)-*condition* if:

- T is a tree, of diameter at most k, that does not contain e,
- the eccentricity of $a(B)$ in $T[S_B^{x,e}]$ is at most i, and
- the eccentricity of $a(B)$ in $T[S_B^{y,e}]$ is at most j.

Given an edge block B, a subgraph T of G_B, and an integer i, we say that T satisfies the (i)-*condition* if:

- T is a tree of diameter at most k and
- the eccentricity of $a(B)$ in T is at most i.

Let us fix a block B, and an edge e of $E(B)$. In the sequel our goal is to describe how to compute $\lambda_B(e)$. We can assume that for every child C of B, the function λ_C has already been computed and C contains at least one edge e' such that $\lambda_C(e') \neq \perp$, since otherwise the algorithm would have stopped.

We define T^e to be the tree obtained by taking the union of all the following:

- the graph $B - e$ if B is a cycle block,
- the graph B if B is an edge block, and
- $\lambda_C(e_C)$ for every child C of B that is an edge block containing only the edge e_C.

For every child C of B that is a cycle block, for every edge e' of C such that $\lambda_C(e') \neq \perp$ and for $x' \in e'$, the tree $R_C^{x',e'}$ is $\lambda_C(e')[S_B^{x',e'}]$. Note that, given a child C of B that is a cycle block, and three vertices v, v', v'' of $V(C)$ such that $v \neq v''$, $v' \neq a(C)$, and $\{v, v'\}$ and $\{v', v''\}$ are in $E(C)$, if $R_C^{v,\{v,v'\}}$ and $R_C^{v',\{v',v''\}}$ are defined, then $R_C^{v,\{v,v'\}}$ is a subgraph of $R_C^{v',\{v',v''\}}$. We define $\mathcal{R}_B = \{R_C^{x',e'} \mid C \in \mathcal{C}(B) \text{ is a cycle block}, e' \in E(C), \lambda_C(e') \neq \perp, x' \in e'\}$.

For $\mathcal{Q} \subseteq \mathcal{R}_B$ we denote by $T_{\mathcal{Q}}^e$ the graph obtained by taking the union of T^e and $\mathcal{Q}$. If there exists $R \in \mathcal{R}_B$ such that $\mathcal{Q} = \{R\}$, we write T_R^e instead of $T_{\mathcal{Q}}^e$. We define $\texttt{close}_{\mathcal{R}_B}(\mathcal{Q})$ to be the set of elements of $\mathcal{R}_B$ that are subgraphs of $T_{\mathcal{Q}}^e$. Note that $T_{\mathcal{Q}}^e = T_{\texttt{close}_{\mathcal{R}_B}(\mathcal{Q})}^e$.

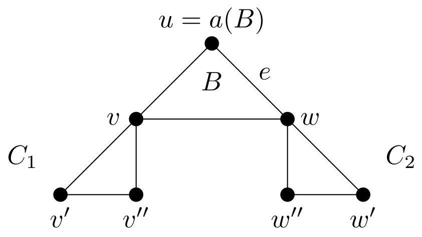

Figure 6 Exemple of a cycle block B with two children C_1 and C_2. When computing $\lambda_B(e)$, we have:

$$\begin{aligned}\phi_0 \;=\; & (\mathbf{v}(R_{C_1}^{v',\{v',v''\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v,\{v,v'\}})) \wedge (\mathbf{v}(R_{C_1}^{v'',\{v'',v\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v',\{v',v''\}})) \wedge \\ & (\mathbf{v}(R_{C_1}^{v'',\{v'',v'\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v,\{v,v''\}})) \wedge (\mathbf{v}(R_{C_1}^{v',\{v',v\}}) \Rightarrow \mathbf{v}(R_{C_1}^{v'',\{v'',v''\}})) \wedge \\ & (\mathbf{v}(R_{C_2}^{w',\{w',w''\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w,\{w,w'\}})) \wedge (\mathbf{v}(R_{C_2}^{w'',\{w'',w\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w',\{w',w''\}})) \wedge \\ & (\mathbf{v}(R_{C_2}^{w'',\{w'',w'\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w,\{w,w''\}})) \wedge (\mathbf{v}(R_{C_2}^{w',\{w',w\}}) \Rightarrow \mathbf{v}(R_{C_2}^{w'',\{w'',w''\}})),\end{aligned}$$

$$\begin{aligned}\phi_1 \;=\; & (\mathbf{v}(R_{C_1}^{v',\{v',v''\}}) \vee \mathbf{v}(R_{C_1}^{v',\{v',v\}})) \wedge (\overline{\mathbf{v}(R_{C_1}^{v',\{v',v''\}})} \vee \overline{\mathbf{v}(R_{C_1}^{v',\{v',v\}})}) \wedge \\ & (\mathbf{v}(R_{C_1}^{v'',\{v'',v'\}}) \vee \mathbf{v}(R_{C_1}^{v'',\{v'',v\}})) \wedge (\overline{\mathbf{v}(R_{C_1}^{v'',\{v'',v'\}})} \vee \overline{\mathbf{v}(R_{C_1}^{v'',\{v'',v\}})}) \wedge \\ & (\mathbf{w}(R_{C_2}^{w',\{w',w''\}}) \vee \mathbf{w}(R_{C_2}^{w',\{w',w\}})) \wedge (\overline{\mathbf{w}(R_{C_2}^{w',\{w',w''\}})} \vee \overline{\mathbf{w}(R_{C_2}^{w',\{w',w\}})}) \wedge \\ & (\mathbf{w}(R_{C_2}^{w'',\{w'',w'\}}) \vee \mathbf{w}(R_{C_2}^{w'',\{w'',w\}})) \wedge (\overline{\mathbf{w}(R_{C_2}^{w'',\{w'',w'\}})} \vee \overline{\mathbf{w}(R_{C_2}^{w'',\{w'',w\}})}), \text{ and}\end{aligned}$$

for every two vertices of $V(C_1) \cup V(C_2)$, say v' and w'', the clause $\overline{\mathbf{v}(R_{C_1}^{v',\{v',v''\}})} \vee \overline{\mathbf{v}(R_{C_2}^{w'',\{w'',w'\}})}$ is a clause of ϕ_2 if and only if the path defined by v', v, w, w'' has diameter greater than k. In the general case, the clauses deal with $T^e_{\{R_{C_1}^{v',\{v',v''\}}, R_{C_2}^{w'',\{w'',w'\}}\}}$ instead of the path v', v, w, w'', but the main idea behind the clauses is the same.

If B is a cycle block, we define for each $i, j \in [0, k]$ the set $\mathcal{R}_B^{(e,i,j)} = \{R \in \mathcal{R}_B \mid T^e_R \text{ satisfies the } (e, i, j)\text{-condition}\}$. If B is an edge block, we define for each $i \in [0, k]$ the set $\mathcal{R}_B^{(i)} = \{R \in \mathcal{R}_B \mid T^e_R \text{ satisfies the } (i)\text{-condition}\}$.

Note that, if B is a cycle block (resp. an edge block), then for each $i, j \in [0, k]$ and for each $R_1, R_2 \in \mathcal{R}_B$ such that R_2 is a subtree of R_1, then if $R_2 \notin \mathcal{R}_B^{(e,i,j)}$ (resp. $R_2 \notin \mathcal{R}_B^{(i)}$) , we have $R_1 \notin \mathcal{R}_B^{(e,i,j)}$ (resp. $R_1 \notin \mathcal{R}_B^{(i)}$).

We associate a boolean variable $\mathbf{v}(R) = \mathbf{v}_C^{x',e'}$ with each $R = R_C^{x',e'} \in \mathcal{R}_B$. With a slight abuse of notation, we say that a set $\mathcal{Q} \subseteq \mathcal{R}_B$ satisfies a formula ϕ over these variables if ϕ is satisfied when each variable of $\{\mathbf{v}(R) \mid R \in \mathcal{Q}\}$ is set to true and each variable of $\{\mathbf{v}(R) \mid R \in \mathcal{R}_B \setminus \mathcal{Q}\}$ is set to false simultaneously. In the following we are going to build three formulas ϕ_0, ϕ_1, and ϕ_2, and if $\mathcal{Q} \subseteq \mathcal{R}_B$ satisfies $\phi_0 \wedge \phi_1 \wedge \phi_2$ then this implies that $T^e_\mathcal{Q}$ is a correct value for $\lambda_B(e)$. Along with the description, the reader is referred to Figure 6 to get some intuition about the formulas ϕ_0, ϕ_1, and ϕ_2.

We construct a 2-Sat formula ϕ_0 such that for each $R_1, R_2 \in \mathcal{R}_B$ where R_2 is a subgraph of R_1, ϕ_0 contains the clause $\mathbf{v}(R_1) \Rightarrow \mathbf{v}(R_2)$. It is easy to show that given $\mathcal{Q} \subseteq \mathcal{R}_B$, $\mathcal{Q}$ satisfies ϕ_0 if and only if $\mathcal{Q} = \texttt{close}_{\mathcal{R}_B}(\mathcal{Q})$.

We construct a 2-Sat formula ϕ_1 as follows. For every child C of B that is a cycle block, and every two consecutive edges $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_2, v_3\}$ of C such that $a(C) \notin \{v_1, v_2\}$ and $R_C^{v_2,e_1}, R_C^{v_2,e_2} \in \mathcal{R}_B$, we add to ϕ_1 two clauses $\mathbf{v}_C^{v_2,e_1} \vee \mathbf{v}_C^{v_2,e_2}$ and

$\overline{\mathbf{v}_C^{v_2,e_1}} \vee \overline{\mathbf{v}_C^{v_2,e_2}}$. With this definition of ϕ_1, we now state the following lemma.

▶ **Lemma 8.** *Let $\mathcal{Q}$ be a subset of $\mathcal{R}_B$ such that $\mathcal{Q}$ satisfies ϕ_0. $\mathcal{Q}$ satisfies ϕ_1 if and only if $T^e_{\mathcal{Q}}$ is a spanning tree of G_B.*

Proof. Let $\mathcal{Q} \subseteq \mathcal{R}_B$. First assume that $T^e_{\mathcal{Q}}$ is a spanning tree of G_B. Let $C \in \mathcal{C}(B)$ be a cycle block, and let $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_2, v_3\}$ be two consecutive edges of C such that $e_1 \neq e_2$, $v_1 \neq a(C)$, and $v_2 \neq a(C)$. As $T^e_{\mathcal{Q}}$ is connected, the clause $\mathbf{v}_C^{v_2,e_1} \vee \mathbf{v}_C^{v_2,e_2}$ is satisfied. As $T^e_{\mathcal{Q}}$ does not contain any cycle, the clause $\overline{\mathbf{v}_C^{v_2,e_1}} \vee \overline{\mathbf{v}_C^{v_2,e_2}}$ is satisfied.

Assume now that $\mathcal{Q}$ satisfies ϕ_1, and let z be a vertex of G_B. If $z \in V(B)$, then there is a path from z to $a(B)$ in T^e and hence also in $T^e_{\mathcal{Q}}$. Otherwise, let $C_z \in \mathcal{C}(B)$ be the block such that $z \in V(G_{C_z})$, and let $s(z)$ be the only vertex of $V(C_z)$ such that $s(z) \neq a(C_z)$ and each path in G_B from $a(B)$ to z contains $s(z)$. Note that if $z \in V(C_z)$, then $s(z) = z$. If C_z is an edge block, then $z \in V(T^e)$; therefore, there is a path from $a(B)$ to z in $T^e_{\mathcal{Q}}$. Otherwise, if C_z is a cycle block, then the condition $\mathbf{v}_C^{v_2,e_1} \vee \mathbf{v}_C^{v_2,e_2}$, with $v_2 = s(z)$, ensures that $z \in T^e_{\mathcal{Q}}$ and that there is a path in $T^e_{\mathcal{Q}}$ from $a(B)$ to z. Thus, $T^e_{\mathcal{Q}}$ is connected and $V(T^e_{\mathcal{Q}}) = V(G_B)$. We need to show that $T^e_{\mathcal{Q}}$ does not contain any cycle. By construction of $T^e_{\mathcal{Q}}$, if it contains a cycle, this cycle should be C where $C \in \mathcal{C}(B)$. The condition $\overline{\mathbf{v}_C^{v_2,e_1}} \vee \overline{\mathbf{v}_C^{v_2,e_2}}$ ensures that C is not a subgraph of $T^e_{\mathcal{Q}}$. ◀

We build a formula ϕ_2 over the variables $\{\mathbf{v}(R) \mid R \in \mathcal{R}_B^{(e,k,k)}\}$. For each $R_1, R_2 \in \mathcal{R}_B^{(e,k,k)}$, $R_1 \neq R_2$, if $T^e_{\{R_1,R_2\}}$ has diameter greater than k, then we add the clause $\overline{\mathbf{v}(R_1)} \vee \overline{\mathbf{v}(R_2)}$ to ϕ_2. With this definition of ϕ_2, we now state the following lemma.

▶ **Lemma 9.** *Let B be a cycle block (resp. an edge block), i and j be two integers of $[0,k]$, and $\mathcal{Q}$ be a subset of $\mathcal{R}_B^{(e,i,j)}$ (resp. of $\mathcal{R}_B^{(i)}$) such that $\mathcal{Q}$ satisfies ϕ_0 and ϕ_1. $\mathcal{Q}$ satisfies ϕ_2 and T^e satisfies the (e,i,j)-condition (resp. the (i)-condition) if and only if $T^e_{\mathcal{Q}}$ is a spanning tree of G_B that satisfies the (e,i,j)-condition (resp. the (i)-condition).*

Proof. Assume that B is a cycle block. Let i, j be two integers in $[0,k]$ and let $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$.

First assume that $T^e_{\mathcal{Q}}$ is a spanning tree of G_B that satisfies the (e,i,j)-condition. This directly implies that T^e also satisfies the (e,i,j)-condition. It remains to show that $\mathcal{Q}$ satisfies ϕ_2. For this, assume that there exist R_1 and R_2 in $\mathcal{Q}$ such that $T^e_{\{R_1,R_2\}}$ has diameter more than k. Since $T^e_{\{R_1,R_2\}}$ is a subtree of $T^e_{\mathcal{Q}}$, this implies that $T^e_{\mathcal{Q}}$ also has diameter more than k, which is a contradiction because $T^e_{\mathcal{Q}}$ satisfies the (e,i,j)-condition.

Assume now that $\mathcal{Q}$ satisfies ϕ_2 and T^e satisfies the (e,i,j)-condition. As $\mathcal{Q}$ satisfies ϕ_1, we know by Lemma 8 that $T^e_{\mathcal{Q}}$ is a spanning tree of G_B. As $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$ and T^e satisfies the (e,i,j)-condition, then the eccentricity of $a(B)$ in $T^e_{\mathcal{Q}}[S_B^{x,e}]$ is at most i, and the eccentricity of $a(B)$ in $T^e_{\mathcal{Q}}[S_B^{y,e}]$ is at most j. Indeed, let $z \in S_B^{x,e}$. If $z \in V(B)$ then, as T^e satisfies the (e,i,j)-condition, we have that $\texttt{cost}_{T^e_{\mathcal{Q}}}(a(b), z) \leq i$ If $z \notin V(B)$ then, as $T^e_{\mathcal{Q}}$ is a spanning tree of G_B, we have that there exists $R \in \mathcal{Q}$ such that $z \in V(R)$. By definition of $\mathcal{R}_B^{(e,i,j)}$, we obtain that $\texttt{cost}_{T^e_{\mathcal{Q}}}(a(b), z) \leq i$ The same argument applies if $z \in S_B^{y,e}$. It remains to show that $T^e_{\mathcal{Q}}$ is of diameter at most k. Let z and z' be two vertices of $T^e_{\mathcal{Q}}$. If both z and z' are in $V(B)$, then as T^e satisfies the (e,i,j)-condition, this implies that $\texttt{cost}_{T^e_{\mathcal{Q}}}(z, z') \leq k$. If $z \in V(B)$ and $z' \notin V(B)$ then, as $T^e_{\mathcal{Q}}$ is a spanning tree of G_B, there exists $R' \in \mathcal{Q}$ such that $z' \in V(R')$. As $R' \in \mathcal{R}_B^{(e,i,j)}$, $\texttt{cost}_{T^e_{\mathcal{Q}}}(z, z') \leq k$. Otherwise, if both z and z' are not in $V(B)$, since $T^e_{\mathcal{Q}}$ is a spanning tree of G_B, there exist $R, R' \in \mathcal{Q}$ such that $z \in V(R)$ and $z' \in V(R')$. If $R = R'$, then as $R' \in \mathcal{R}_B^{(e,i,j)}$, $\texttt{cost}_{T^e_{\mathcal{Q}}}(z, z') \leq k$. Otherwise, as $\mathcal{Q}$ satisfies ϕ_2, then $T^e_{\{R,R'\}}$ has diameter at most k; therefore, $\texttt{cost}_{T^e_{\mathcal{Q}}}(z, z') \leq k$.

The same arguments apply if B is an edge block. ◀

▶ **Lemma 10.** *If B is a cycle block (resp. an edge block) and if there exists a spanning tree $\hat{T}_B$ of that satisfies the (e,i,j)-condition (resp. (i)-condition) for some $i,j \in [0,k]$, then there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2.*

Proof. For readability, we consider the case where B is an edge block. Let $x = a(B)$. Assume that there exists $\hat{T}_B$, a spanning tree of G_B, that satisfies the (i)-condition for some $i \in [0,k]$. We define $\mathcal{Q} = \texttt{close}_{\mathcal{R}_B}(\{R_C^{x',e'} \mid C \in \mathcal{C}(B), C$ is a cycle block, $e' \in E(C), e' \notin E(\hat{T}_B[V(C)]), x' \in e'\})$ and we claim that $\mathcal{Q}$ satisfies ϕ_0, ϕ_1, and ϕ_2. By definition of $\texttt{close}_{\mathcal{R}_B}$, $\mathcal{Q}$ satisfies ϕ_0. It is not difficult to see that $T_{\mathcal{Q}}^e$ is a spanning tree and so, by Lemma 8, $\mathcal{Q}$ satisfies ϕ_1. Let z be a vertex of $V(G_B) \setminus V(B)$, and let $C \in \mathcal{C}(B)$ such that $z \in V(G_C)$. The path in $\hat{T}_B$ and the path in $T_{\mathcal{Q}}^e$ from $a(B)$ to z use exactly the same edges of C. This implies that $\texttt{cost}_{T_{\mathcal{Q}}^e}(a(B),z) \leq i$. Otherwise $\hat{T}_B[V(G_C)]$ would have been a better value for $\lambda_C(e')$ for the only compatible edge $e' \in E(C)$. With the same arguments we show that $T_{\mathcal{Q}}^e$ has diameter at most k. This implies that $T_{\mathcal{Q}}^e$ satisfies the (i)-condition, and so $\mathcal{Q} \subseteq \mathcal{R}_B^{(i)}$ and $\mathcal{Q}$ satisfies ϕ_2.

The same arguments also work if B is a cycle block but we should take care about the part that is in $S_B^{x,e}$ and the part that is in $S_B^{y,e}$ separately. ◀

We now have all the elements to compute the value $\lambda_B(e)$. We assume that B is a cycle block (resp. an edge block). If there is no $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,k,k)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(k)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2, or T^e does not satisfy the (e,k,k)-condition (resp. (k)-condition), then we set $\lambda_B(e) = \bot$. Otherwise, we aim at computing two integers i_0 and j_0 that are the smallest i and j in $[0,k]$ such that there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i,j)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2 and such that T^e satisfies the (e,i,j)-condition (resp. (i)-condition). In order to compute i_0 and j_0, we first fix j to be k and do a binary search on i, between 0 and k, to find the smallest value i_0 such that there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i_0,k)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i_0)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2 and such that T^e satisfies the (e,i_0,k)-condition (resp. (i_0)-condition). We fix this value of i_0 and we do a second binary search, this time on j, between 0 and k, to find the smallest value j_0 such that there exists $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i_0,j_0)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i_0)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2 and such that T^e satisfies the (e,i_0,j_0)-condition (resp. (i_0)-condition). We fix this value of j_0 and we also fix $\mathcal{Q} \subseteq \mathcal{R}_B^{(e,i_0,j_0)}$ (resp. $\mathcal{Q} \subseteq \mathcal{R}_B^{(i_0)}$) that satisfies ϕ_0, ϕ_1, and ϕ_2. We set $\lambda_B(e) = T_{\mathcal{Q}}^e$. Using Lemma 8 and Lemma 9, we know that the graph $\lambda_B(e)$ is a spanning tree of G_B that satisfies the (e,i_0,j_0)-condition (resp. (i_0)-condition). By Lemma 10, there is no spanning subtree of G_B that satisfies the (e,i_1,j_1)-condition (resp. (i_1)-condition) with $i_1 < i_0$ or $j_1 < j_0$ (resp. $i_1 < i_0$). This finishes the description of the algorithm.

Let us now discuss about the running time of the algorithm. At each step, given a cycle block (resp. an edge block) B and $e \in E(B)$, for each $i,j \in [0,k]$, we can check if T^e satisfies the (e,i,j)-condition (resp. the (i)-condition) in time $\mathcal{O}(n^2)$. Moreover, the number of elements in $\mathcal{R}_B$ is linear in n and for each $i,j \in [0,k]$, $\mathcal{R}_B^{(e,i,j)}$ can be computed in time $\mathcal{O}(n^2)$. As $\mathcal{R}_B$ contains at most $\mathcal{O}(n)$ elements, then the 2-Sat formulas ϕ_0, ϕ_1, and ϕ_2 contain at most $\mathcal{O}(n^2)$ clauses. We can check for each of the $\mathcal{O}(n^2)$ possible clauses if it is in ϕ_0, ϕ_1, or ϕ_2 in time $\mathcal{O}(n)$. Hence, we can compute ϕ_0, ϕ_1 and ϕ_2 in time $\mathcal{O}(n^3)$. As they contain at most $\mathcal{O}(n^2)$ clauses, we can solve them in time $\mathcal{O}(n^2)$. Since for each block B and each edge $e \in E(B)$, we perform at most two (independent) binary searches to find i_0 and j_0, we can compute $\lambda_B(e)$ in time $\mathcal{O}(n^3 \cdot \log k)$. Because there is a linear number of values $\lambda_B(e)$ to compute, we obtain an algorithm that solves Diameter-Tree* in time $\mathcal{O}(n^4 \cdot \log k)$. Using again a binary search on k between 0 and $2^{\lceil \log \mathsf{opt} \rceil}$, and the previous algorithm that solves Diameter-Tree* as a subroutine, we obtain an algorithm that solves Diameter-Tree in time $\mathcal{O}(n^4 \cdot (\log \mathsf{opt})^2)$ where opt is the diameter of the solution.

E Proof of Theorem 5

Before proceeding to the description of the algorithm, we first need some definitions.

Nice tree decompositions. Let $\mathcal{D} = (Y, \mathcal{X})$ be a tree decomposition of G, r be a vertex of Y, and $\mathcal{G} = \{G_t \mid t \in V(Y)\}$ be a collection of subgraphs of G, indexed by the vertices of Y. We say that the triple $(\mathcal{D}, r, \mathcal{G})$ is *nice* if the following conditions hold:

- $X_r = \emptyset$ and $G_r = G$,
- each node of $\mathcal{D}$ has at most two children in Y,
- for each leaf $t \in V(Y)$, $X_t = \emptyset$ and $G_t = (\emptyset, \emptyset)$. Such a t is called a *leaf node*,
- if $t \in V(\mathcal{T})$ has exactly one child t', then either
  - $X_t = X_{t'} \cup \{v_{\text{insert}}\}$ for some $v_{\text{insert}} \notin X_{t'}$ and $G_t = (V(G_{t'}) \cup \{v_{\text{insert}}\}, E(G_{t'}))$. The node t is called *vertex-introduce node* and the vertex 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. The node t is called *edge-introduce node* and the edge e_{insert} is the *insertion edge* of X_t, or
  - $X_t = X_{t'} \setminus \{v_{\text{forget}}\}$ for some $v_{\text{forget}} \in X_{t'}$ and $G_t = G_{t'}$. The node t is called *forget node* and v_{forget} is the *forget vertex* of X_t.
- if $t \in V(Y)$ has exactly two children t' and t'', then $X_t = X_{t'} = X_{t''}$, and $E(G_{t'}) \cap E(G_{t''}) = \emptyset$. The node t is called a *join node.*

The notion of a nice triple defined above is essentially the same as the one of nice tree decomposition in [10] (which in turn is an enhancement of the original one, introduced in [27]). As already argued in [10, 27], it is possible, given a tree decomposition to transform it in polynomial time to a new one $\mathcal{D}$ of the same width and construct a collection $\mathcal{G}$ such that the triple $(\mathcal{D}, r, \mathcal{G})$ is nice.

Transfer triples and their fusion. Let (F, R, α) be a triple where F is a forest, $R \subseteq V(F)$, and $\alpha : R \times R^F \to [0, k] \cup \{\perp\}$, where $R^F = V(F) \cup (E(F) \setminus \mathsf{adj}_F(R))$. Keep in mind that R^F contains all vertices and edges of F except from the edges that are incident to vertices in R. We call (F, R, α) a *transfer triple* if, given a $(v, a) \in R \times R^F$, $\alpha(v, a) = \perp$ if and only if v and a belong in different connected components of F. The function α will be used for indicating for each pair (v, a) the "cost of transfering" from v to a in F (α is not necessarily a distance function).

Let (F_1, R_1, α_1) and (F_2, R_2, α_2) be two transfer triples where $R = R_1 = R_2$, $E(F_1) \cap E(F_2) = \emptyset$, and such that $F = F_1 \cup F_2$ is a forest. Let also $\beta : \mathsf{adj}_{F_1}(R) \times \mathsf{adj}_{F_2}(R) \to [0, k] \cup \{\perp\}$. We require a function $\alpha_1 \oplus_\beta \alpha_2 : R \times R^F \to [0, k] \cup \{\perp\}$ 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 $(v, a) \in R \times R^F$. Let P be the shortest path in F containing 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

$$\alpha_1 \oplus_\beta \alpha_2(v,a) \;=\; \sum_{h\in[0,\lfloor\frac{q-1}{2}\rfloor]} \alpha_1(v_{2i_h}, v_{2i_h+1}) + \sum_{h\in[0,\lfloor\frac{q-2}{2}\rfloor]} \alpha_2(v_{2i_h+1}, v_{2i_h+2}) + \sum_{h\in[q]} \beta(e^-_{i_h}, e^+_{i_h}) + \alpha_{(q \bmod 2)+1}(v_{i_q}, a).$$

Before we start the description of the dynamic programming we give some more definitions. Let F be a forest and let S be a set of vertices in F that is good for G. 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:

- removing a vertex of degree 1 and
- 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 $V(F') \setminus \{x\}$. 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 $1 \leq |K_z| \leq 2$ 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 that $|K_z| \geq 1$. 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, then P is the unique path from x_1 to x_2 in F. Let 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 $|K_z| \leq 2$.

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*. 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. $|V(F) \setminus N_F[X_t]| \leq w_t - 2$, and
  5. $|\{e \in E(F) \mid e \cap X_t = \emptyset\}| \leq 2w_t - 3$,
- $\alpha : X_t \times X_t^F \to [0,k] \cup \{\perp\}$,

We call the vertices in $V(F) \setminus N_F[X_t]$ *external* vertices of F and the edges of $\{e \in E(F) \mid e \cap X_t = \emptyset\}$ *external* edges of F.

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 $\beta_t(e_1, e_2) = \perp$.

Let (F, α) be a t-pair. Recall that X_t^F contains all t-terminals and all non-t-terminal edges of F. 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 $\alpha(a, a') \leq k$,
- some internal vertex b of the path P between a and a' in F belongs in X_t and $\alpha_t(b, a) + \beta_t(e^-, e^+) + \alpha_t(b, a') \leq k$, 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.

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 $\mathsf{diam}(T) \leq k$ 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 $\alpha(x, y) \geq \mathtt{cost}_{\hat{F}}(x, z)$.

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 internal vertices (resp. edges) of a tree with no vertices of degree 2 are at most two less than the number of its leaves (resp. at most twice the number of its 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 4. For every $t \in V(Y)$, it holds that $|\mathcal{R}_t| \leq k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$.

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 forest 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 function α is at most $k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)}$. So $|\mathcal{R}_t| \leq k^{\mathcal{O}(\Delta \cdot \mathsf{tw}^2)} \cdot (\Delta \cdot \mathsf{tw})^{\mathcal{O}(\mathsf{tw})}$ and the claim holds. ◄

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 descendent 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 $\mathcal{R}_t = \{((\emptyset, \emptyset), \varnothing)\}$.
- 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

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

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
$$\begin{aligned} R_t \;=\; & R_{t'} \cup \{(F, \alpha) \mid (F, \alpha) \text{ is admissible, } F \text{ is a forest, and there exists a pair} \\ & (F', \alpha') \in R_{t'} \text{ such that } F = F' \cup F'' \text{ and } \alpha = \alpha' \oplus_{\beta_t} \alpha''\}. \end{aligned}$$
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 to include 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 an other 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
$$\begin{aligned} R_t \;=\; & R_{t'} \cup \{(F, \alpha) \mid (F, \alpha) \text{ is admissible, } F \text{ is a forest, and there exist two} \\ & \text{pairs } (F', \alpha') \in R_{t'} \text{ and } (F'', \alpha'') \in R_{t''} \text{ such} \\ & \text{that } F = F' \cup F'' \text{ and } \alpha = \alpha' \oplus_{\beta_t} \alpha''\}. \end{aligned}$$
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 4 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.

F Proof of Claim 3

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 $\mathsf{diam}(T) \leq 2B$ 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 $\mathsf{diam}(T) \leq 2B$.

Conversely, let T be a spanning tree of G with $\mathsf{diam}(T) \leq 2B$, 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 $\mathsf{diam}(T) \leq 2B$, for every $i \in [n]$ and every $x \in \{1, 2\}$, tree T_x contains the two edges $\{v_i, \ell^i_j\}$ and $\{v_i, r^i_j\}$ 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 $\mathsf{diam}(T) \geq \frac{1}{2}\mathsf{diam}(T_1) + \frac{1}{2}\mathsf{diam}(T_2)$, and since $\mathsf{diam}(T) \leq 2B$, it follows that there exists $x \in \{1, 2\}$ such that $\mathsf{diam}(T_x) \leq B$. Assume without loss of generality that $x = 1$, i.e., that $\mathsf{diam}(T_1) \leq B$. We define the bins $S_1, \ldots, S_k$ as follows. For every $i \in [n]$, if T_1 contains the two edges $\{v_i, \ell^i_j\}$ and $\{v_i, r^i_j\}$, 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 $\mathsf{diam}(T_1) \leq B$.