

RESEARCH

Open Access



# Tree diet: reducing the treewidth to unlock FPT algorithms in RNA bioinformatics

Bertrand Marchand<sup>1,2</sup>, Yann Ponty<sup>1\*</sup> and Laurent Bulteau<sup>2\*</sup>

## Abstract

Hard graph problems are ubiquitous in Bioinformatics, inspiring the design of specialized Fixed-Parameter Tractable algorithms, many of which rely on a combination of tree-decomposition and dynamic programming. The time/space complexities of such approaches hinge critically on low values for the treewidth  $tw$  of the input graph. In order to extend their scope of applicability, we introduce the TREE-DIET problem, *i.e.* the removal of a minimal set of edges such that a given tree-decomposition can be slimmed down to a prescribed treewidth  $tw'$ . Our rationale is that the time gained thanks to a smaller treewidth in a parameterized algorithm compensates the extra post-processing needed to take deleted edges into account. Our core result is an FPT dynamic programming algorithm for TREE-DIET, using  $2^{O(tw)}n$  time and space. We complement this result with parameterized complexity lower-bounds for stronger variants (e.g., NP-hardness when  $tw'$  or  $tw - tw'$  is constant). We propose a prototype implementation for our approach which we apply on difficult instances of selected RNA-based problems: RNA design, sequence-structure alignment, and search of pseudoknotted RNAs in genomes, revealing very encouraging results. This work paves the way for a wider adoption of tree-decomposition-based algorithms in Bioinformatics.

**Keywords:** RNA, treewidth, FPT algorithms, RNA design, structure sequence alignment

## Introduction

Graph models and parameterized algorithms are found at the core of a sizable proportion of algorithmic methods in bioinformatics addressing a wide array of subfields, spanning sequence processing [1], structural bioinformatics [2], comparative genomics [3], phylogenetics [4], and further examples that can be found in a review by Bulteau and Weller [5]. RNA bioinformatics is no exception, with the prevalence of the secondary structure, an outer planar graph [6], as an abstraction of RNA conformations, and the notable utilization of graph models to represent complex topological motifs called pseudo-knots [7], inducing the hardness of several tasks, such as structure prediction [8–10], structure alignment [11], or

structure/sequence alignment [12]. Such motifs are functionally important and conserved, as witnessed by their presence in the consensus structure of 336 RNA families in the 14.5 edition of the RFAM database [13]. Moreover, methods in RNA bioinformatics [14] are increasingly considering non-canonical base pairs and modules [15, 16], further increasing the density of RNA structural graphs and outlining the need for scalable algorithms.

A parameterized complexity approach can be used to circumvent the frequent NP-hardness of relevant problems. It generally considers one or several parameters, whose values are naturally bounded (or much smaller than the input size) within real-life instances. Once relevant parameters have been identified, one aims to design a Fixed Parameter Tractable (FPT) algorithm, having polynomial complexity for any fixed value of the parameter, and reasonable dependency on the parameter value. The treewidth is a classic parameter for FPT algorithms, and intuitively captures a notion of distance of the input to a tree. It is popular in bioinformatics due to the existence

\*Correspondence: [yann.ponty@lix.polytechnique.fr](mailto:yann.ponty@lix.polytechnique.fr); [laurent.bulteau@u-pem.fr](mailto:laurent.bulteau@u-pem.fr)

<sup>1</sup> LIX CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

<sup>2</sup> LIGM, CNRS, Univ Gustave Eiffel, 77454 Marne-la-Vallée, France



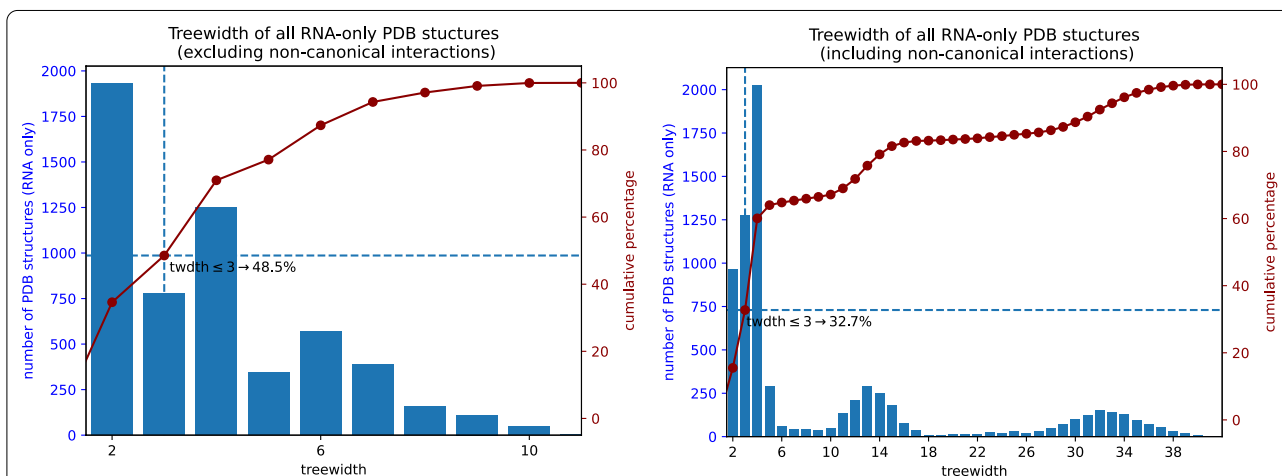
of efficient heuristics [17, 18] for computing tree-decompositions of reasonable treewidth. Given a tree-decomposition, many combinatorial optimization tasks can be solved using dynamic programming (DP), in time/space complexities that remain polynomial for any fixed treewidth value. Resulting algorithms remain correct upon (almost) arbitrary modifications of the objective function parameters, and can be adapted to study statistical properties of search spaces through changes of algebra.

Unfortunately, the existence of a parameterized (or FPT) algorithm does not necessarily imply that of a practically-efficient implementation, even when the parameter takes low typical values. Indeed, the dependency of the complexity on the treewidth may be prohibitive, both in terms of time and memory requirements. This limitation is particularly obvious while searching and aligning structured RNAs, giving rise to an algorithmic problem called RNA structure-sequence alignment [12, 19, 20], for which the best known exact algorithm is in  $\Theta(n \cdot m^{tw+1})$ , with  $n$  the structure length,  $m$  the sequence/window length, and  $tw$  the treewidth of the structure (inc. backbone). Such a complexity becomes impractical for structures having a treewidth higher than  $\sim 4$ , which represent 50 to 70% of known RNA structures, as shown by Fig. 1, based on a broad analysis of structures found in the PDB database. Similar complexities hold for problems that can be expressed as (weighted) constraint satisfaction problems, with  $m$  representing the cardinality of the variable domains. Such frameworks are frequently used for molecular design, both in proteins [21] and RNA

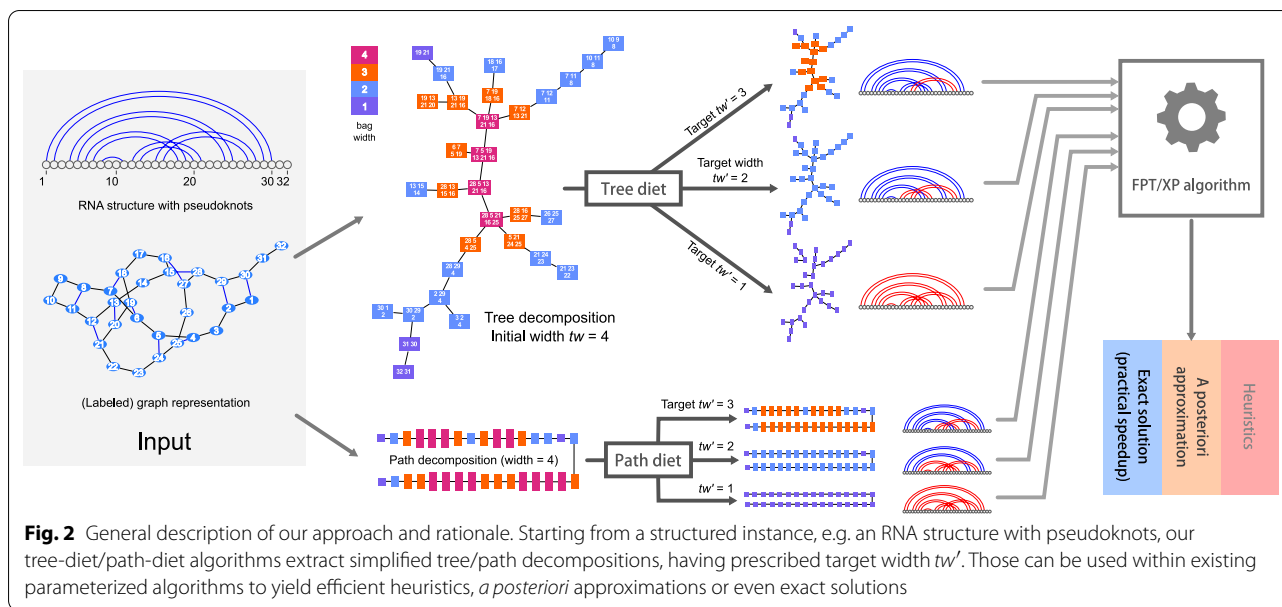
[22], and may require the consideration of tree-widths as high as 20 or more [23].

In this paper, we investigate a pragmatic strategy to increase the practicality of parameterized algorithms based on the treewidth parameter [27]. We put our instance graphs on a diet, i.e. we introduce a preprocessing that reduces their treewidth to a prescribed value by removing a minimal cardinality set of edges. As discussed previously, the practical complexity of many algorithms greatly benefits from the consideration of simplified instances, having lower treewidth. Moreover, specific countermeasures for errors introduced by the simplification can sometimes be used to preserve the correctness of the algorithm. For instance, for searching structured RNAs using RNA structure-sequence alignment [19], an iterated filtering strategy could use instances of increasing treewidth to restrict potential hits, weeding them early so that a—costly—full structure is reserved to (quasi-)hits. This strategy could remain exact while saving substantial time. Alternative countermeasures could be envisioned for other problems, such as a rejection approach to correct a bias introduced by simplified instances in RNA design. An overview of our approach is sketched on Fig. 2

After stating our problem(s) in Sect. 2, we study in Sect. 3 the parameterized complexity of the GRAPH-DIET problem, the removal of edges to reach a prescribed treewidth. We propose, in Sect. 4, a practical Dynamic Programming FPT algorithm for TREE-DIET, along with possible further optimizations for PATH-DIET,



**Fig. 1** Histogram of treewidth values over all RNA-only structures in the PDB database [24]. The data consists of 5 760 non-redundant graphs, each corresponding to a “chain” of a PDB entity. The nucleotide chains and their base pairs were extracted using the DSSR tool [25]. On each of these graphs, 4 standard treewidth heuristics from the LibTW library [26] (min-degree, min-fill-in, lex-BFS, max-cardinality-search) were launched, and the best width result was selected. Even if these heuristics reputedly tend to yield results close to the optimal, these results are still upper bounds. For each individual structure, the actual treewidth value may be lower. Depending on whether non-canonical base pairs are taken into account (right) or not (left), the proportion of structures having a width  $\geq 4$  ranges from 50 to 70%. For such values, the complexity of structure-sequence alignment ( $O(n \cdot m^{tw+1})$ ) becomes prohibitive. It is also worth noting that only *pseudo-knotted* structures may have a treewidth  $\geq 3$



two natural simplifications of the GRAPH-DIET problem, where a tree (resp. path) decomposition is provided as input and used as a guide. Finally, we show in Sect. 5 how our algorithm can be used to extract hierarchies of graphs/structural models of increasing complexity to provide alternative sampling strategies for RNA design, and speed-up the search for pseudoknotted non-coding RNAs. We conclude in Sect. 6 with future considerations and open problems.

**Statement of the problem(s) and results**

A *tree-decomposition*  $\mathcal{T}$  (over a set  $V$  of vertices) is a tree whose nodes are subsets of  $V$ , known as bags. The bags containing any  $v \in V$  induce a (connected) subtree of  $\mathcal{T}$ . A *path-decomposition* is a tree-decomposition whose underlying tree  $\mathcal{T}$  is a path. The *width* of  $\mathcal{T}$  (denoted  $w(\mathcal{T})$ ) is the size of its largest bag minus 1. An edge  $\{u, v\}$  is *visible* in  $\mathcal{T}$  if some bag contains both  $u$  and  $v$ , otherwise it is *lost*.  $\mathcal{T}$  is a *tree-decomposition* of  $G$  if all edges of  $G$  are visible in  $\mathcal{T}$ . The *treewidth* of a graph  $G$  is the minimum width over all tree-decompositions of  $G$ .

**Problem (GRAPH-DIET)** Given a graph  $G = (V, E)$  of treewidth  $tw$ , and an integer  $tw' < tw$ , find a tree-decomposition over  $V$  of width at most  $tw'$  losing a minimum number of edges from  $G$ .

A *tree-diet* of  $\mathcal{T}$  is any tree-decomposition  $\mathcal{T}'$  obtained by removing vertices from the bags of  $\mathcal{T}$ .  $\mathcal{T}'$  is a  $d$ -tree-diet if  $w(\mathcal{T}') \leq w(\mathcal{T}) - d$ .

**Problem (TREE-DIET)** Given a graph  $G$ , a tree-decomposition  $\mathcal{T}$  of  $G$  of width  $tw$ , and an integer  $tw' < tw$ , find a  $(tw - tw')$ -tree-diet of  $\mathcal{T}$  losing a minimum number of edges.

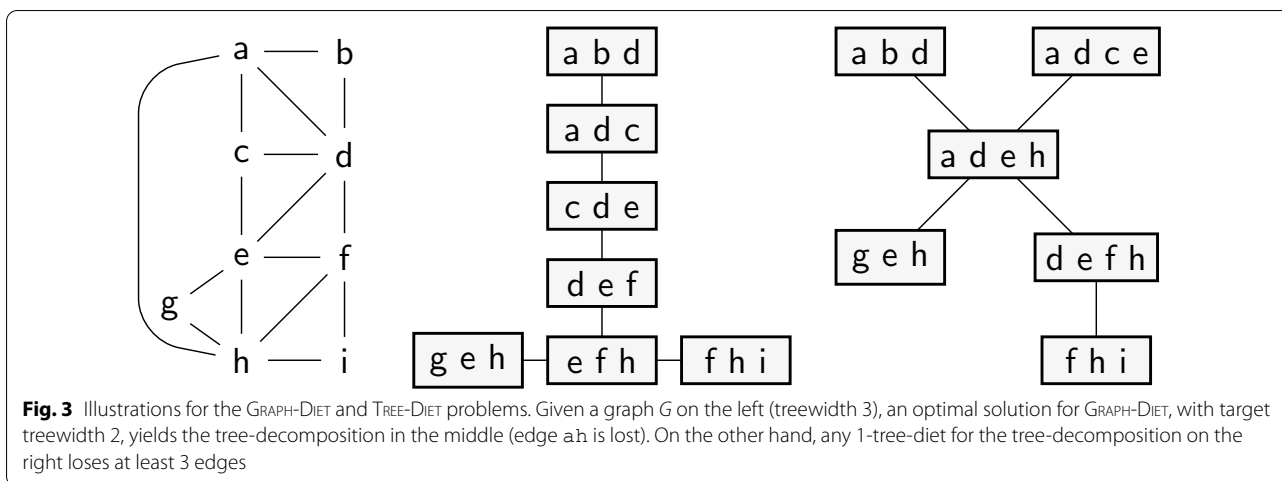
Note that for TREE-DIET,  $\mathcal{T}$  does not have to be optimal, so the width  $tw$  of the input tree decomposition might be larger than the actual treewidth of  $G$ , thus TREE-DIET can be used to reduce the width of *any* input decomposition. We define BINARY-TREE-DIET and PATH-DIET analogously, where  $\mathcal{T}$  is restricted to be a binary tree (respectively, a path). An example of an instance of GRAPH-DIET and of TREE-DIET are given in Fig. 3.

**Parameterized complexity in a nutshell**

The basics of parameterized complexity can be loosely defined as follows (see [28] for the formal background). A *parameter*  $k$  for a problem is an integer associated with each instance which is expected to remain small in practical instances (especially when compared to the input size  $n$ ). An exact algorithm, or the problem it solves, is FPT if it takes time  $f(k)\text{poly}(n)$ , and XP if it takes time  $n^{g(k)}$  (for some functions  $f, g$ ). Under commonly accepted conjectures (see for instance [29] for details), W[1]-hard problems may not be FPT, and Para-NP-hard problems (NP-hard even for some fixed value of  $k$ ) are not FPT nor XP.

**Our results**

Our results are summarized in Table 1. Although the GRAPH-DIET problem would give the most interesting



**Table 1** Parameterized results for our problems. Algorithm complexities are given up to polynomial time factors ( $O^*$  notation),  $\Delta$  denotes the maximum number of children in the input tree-decomposition

Parameter	Source treewidth	Target treewidth	Difference
Problem	$tw$	$tw'$	$d = tw - tw'$
GRAPH-DIET	FPT via MSO Theorem 1	Para-NP-hard $tw' = 2$ EDP( $K_4$ ) [30]	Para-NP-hard <sup>a</sup> $d = 1$ Theorem 3
TREE-DIET	XP $O^*((6\Delta)^{tw})$ Theorem 7	FPT open Para-NP-hard $tw' = 1$ Theorem 3	Para-NP-hard $d = 1$ Theorem 5
BINARY-TREE-DIET	FPT		W[1]-hard Theorem 5
PATH-DIET	$O^*(12^{tw})$ Theorem 7		XP $O^*(tw^d)$ Theorem 8

<sup>a</sup> See Theorem 2 statement for a more precise formulation

tree-decompositions in theory, it seems unlikely to admit efficient algorithms in practice (see Sect. 3).

Thus we focus on the TREE-DIET relaxation, where an input tree-decomposition is given, which we use as a guide/restriction towards a thinner tree-decomposition. Seen as an additional constraint, it makes the problem harder (the case  $tw' = 1$  becomes NP-hard, Theorem 3, although for GRAPH-DIET it corresponds to the SPANNING TREE problem and is polynomial). With parameter  $tw$  however, it does help reduce the search space. In Theorem 7 we give an  $O((6\Delta)^{tw} \Delta^2 n)$  Dynamic Programming algorithm, where  $\Delta$  is the maximum number of children of any bag in the tree-decomposition. This algorithm can thus be seen as XP in general, but FPT on bounded-degree tree-decompositions

(e.g. binary trees and paths). This is not a strong restriction, since the input tree may safely and efficiently be transformed into a binary one (see Supplementary Section A for more details). Moreover, the duplications of bags which are used in the conversion may only decrease the number of lost edges incurred by TREE-DIET.

We also consider the case where the treewidth needs to be reduced by  $d = 1$  only, this without constraining the source treewidth. We give a polynomial-time algorithm for PATH-DIET in this setting (Theorem 8) which generalizes into an XP algorithm for larger values of  $d$ , noting that an FPT algorithm for  $d$  is out of reach by Theorem 5. We also show that the problem is Para-NP-hard if the tree degree is unbounded (Theorem 4).

### Algorithmic limits: parameterized complexity considerations

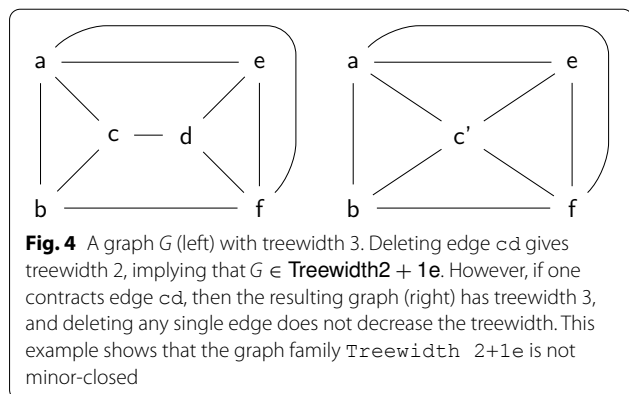
GRAPH-DIET can be seen as a special case of the EDGE DELETION PROBLEM (EDP) for the family of graphs  $\mathcal{H}$  of treewidth  $tw'$  or less: given a graph  $G$ , remove as few edges as possible to obtain a graph in  $\mathcal{H}$ . Such edge modification problems are more often parameterized by the number  $k$  of edited edges (see [31] for a complete survey). Given our focus on increasing the practicality of treewidth-based algorithms in bioinformatics, we restrict our focus to treewidth related parameters  $tw$ ,  $tw'$  and  $d = tw - tw'$ .

Considering the target treewidth  $tw'$ , we note that EDP is NP-hard when  $\mathcal{H}$  is the family of treewidth-2 graphs [30], namely  $K_4$ -free graphs, hence the notation  $EDP(K_4)$ . It follows that GRAPH-DIET is Para-NP-hard for the target treewidth parameter  $tw'$ .

#### Graph-diet: practical solutions seem unlikely

For a combination of the parameters  $tw'$  and  $k$ , we could imagine graph minor theorems yielding parameterized algorithms “for free”, as it is often the case with treewidth-based problems. In this respect, GRAPH-DIET corresponds to deciding if a graph  $G$  belongs to the family of graphs having treewidth  $tw'$ , augmented by  $k$  additional edges, denoted as  $Treewidth\text{-}tw'+k\text{e}$  since its introduction by Cai [32]. If this family were minor-closed, polynomial minor-free-testing [33, 34] would yield an FPT algorithm. However, this is not the case: for some graphs in the family, an edge contraction yields a graph  $G'$  not in  $Treewidth\text{-}tw'+k\text{e}$ , as illustrated by Fig. 4.

Regarding the source graph treewidth  $tw$ , the vertex deletion equivalent of GRAPH-DIET, where one asks for a minimum subset of vertices to remove to obtain a given treewidth, is known as a TREewidth MODULATOR. This problem has been better-studied than its edge-deletion counterpart [35], and has been shown to be FPT for the treewidth [36]. For the edge-deletion



version (GRAPH-DIET), we can use an optimization variant of Courcelle’s Theorem [29, Thm. 7.12] to show that the problem is FPT for  $tw$ . However, this is a purely theoretical result as the running-time of such “black-box” algorithms typically involve towers of exponentials on the treewidth parameter.

**Theorem 1** GRAPH DIET is FPT for the treewidth.

**Proof** We formulate GRAPH DIET as a Monadic Second-Order Logic (MSO) formula as follows: given a graph  $G = (V, E)$ , an integer  $tw'$  and a set  $X$  of edges, let  $\phi_{tw'}(G, X)$  be true iff  $G[E \setminus X]$  has treewidth  $tw'$ . Clearly  $\phi_{tw'}$  can be expressed as an MSO formula, since both  $G[E \setminus X]$  and “being of treewidth  $tw'$ ” can be expressed in MSO [37]. Thus, by Arnborg et al. [38], there exists an algorithm that, given  $G$  of treewidth  $tw$ , finds a set  $X$  of minimum size satisfying  $\phi_{tw'}(G, X)$  in time  $f_{tw'}(tw) \cdot n$ . Writing  $g(tw) = \max_{tw' \leq tw} f_{tw'}(tw)$ , this yields an algorithm for GRAPH DIET running in time at most  $g(tw) \cdot n$ . □

Overall, even though GRAPH DIET is FPT for the treewidth, “practical” exact algorithms seem out of reach. Indeed, any algorithm for GRAPH-DIET can be used to compute the TREewidth of an arbitrary graph, for which current state-of-the-art exact algorithms require time in  $tw^{O(tw^3)}$  [27]. We thus have the following conjecture, which motivates the TREE-DIET relaxation of the problem.

**Conjecture 1** GRAPH-DIET does not admit algorithms with single-exponential running time for the treewidth.

On a related note, it is worth noting that Edge Deletion to other graph classes (interval, permutation, ...) does admit efficient algorithms when parameterized by the treewidth alone [39], painting a contrasted picture.

Finally, for parameter  $d$ , any polynomial-time algorithm for constant  $d$  would allow to compute the treewidth of any graph in polynomial time. Since treewidth is NP-hard we have the following result.

**Theorem 2** There is no XP algorithm for GRAPH-DIET with parameter  $d$  unless  $P = NP$ .

**Proof** We consider the decision version of GRAPH-DIET where a bound  $k$  on the number of deleted edges is given. We build a Turing reduction from TREewidth: more precisely, assuming an oracle for GRAPH-DIET with  $d = 1$  is available, we build a polynomial-time algorithm to compute the treewidth of a graph  $G$ . This is achieved by computing  $GRAPH\text{-}DIET(G, tw, d = 1, k = 0)$  for

decreasing values of  $tw$  (starting with  $tw = |V|$ ): the first value of  $tw$  for which this call returns no solution is the treewidth of  $G$ . Note that this is not a *many-one* reduction, since several calls to GRAPH-DIET may be necessary (so this does not precisely qualify as an NP-hardness reduction, even though a polynomial-time algorithm for GRAPH-DIET( $G, tw, d = 1, k = 0$ ) would imply  $P=NP$ ).  $\square$

**Lower bounds for tree-diet**

Parameters  $tw'$  and  $d$  would be the most interesting in practice, since parameterized algorithms would be efficient for small diets or small target treewidth. However, we prove strong lower-bounds for TREE-DIET on each of these parameters, leaving very little hope for parameterized algorithms (we thus narrow down the possible algorithms to the combined parameter  $tw' + d$ , i.e.  $tw$ , see Sect. 4). Only XP for parameter  $d$  when  $\mathcal{T}$  has a constant degree remains open (cf. Table 1).

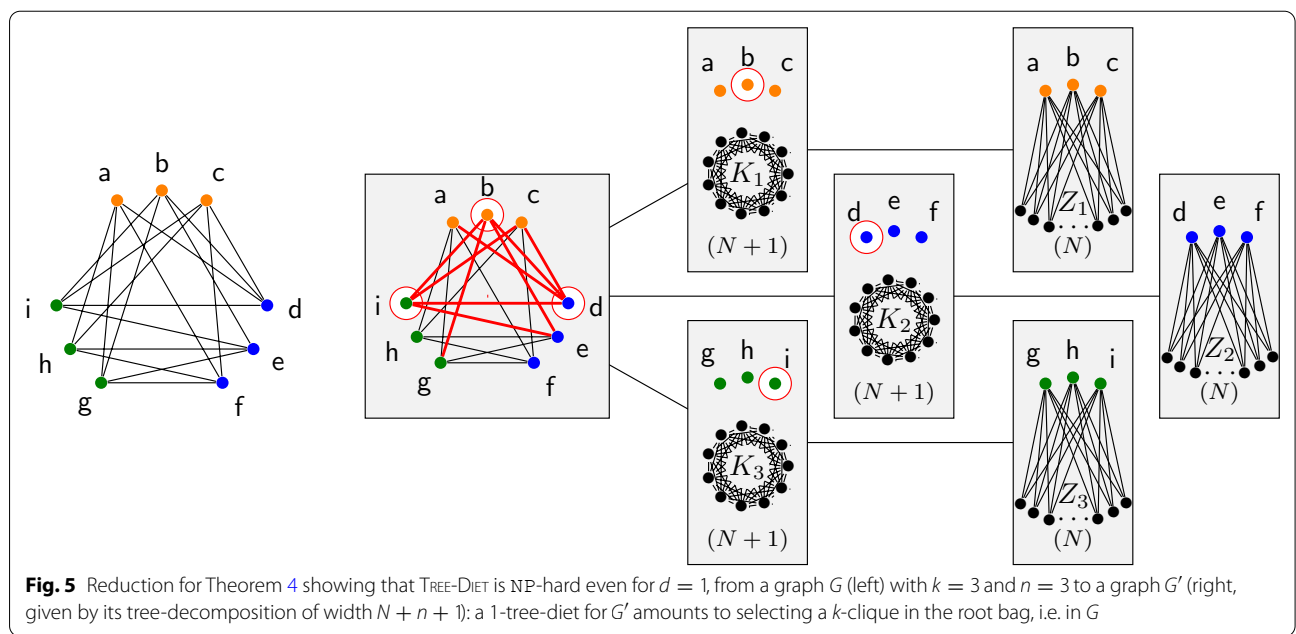
**Theorem 3** TREE-DIET and PATH-DIET are Para-NP-hard for the target treewidth parameter  $tw'$  (NP-hard for  $tw' = 1$ ).

**Proof** By reduction from the NP-hard problem SPANNING CATERPILLAR TREE [40]: given a graph  $G$ , does  $G$  have a spanning tree  $C$  that is a caterpillar? Given  $G = (V, E)$  with  $n = |V|$ , we build a tree-decomposition  $\mathcal{T}$  of  $G$  consisting of  $n - 1$  bags containing all vertices (the width of the decomposition is therefore  $n - 1$ )

connected in a path. Then  $(G, \mathcal{T})$  admits a tree-diet to treewidth 1 with  $n - 1$  visible edges if, and only if,  $G$  admits a caterpillar spanning tree. Indeed, the subgraph of  $G$  with visible edges must be a graph with pathwidth 1, i.e. a caterpillar [41]. With  $n - 1$  visible edges, the caterpillar connects all  $n$  vertices together, i.e. it is a spanning tree.  $\square$

**Theorem 4** TREE-DIET is Para-NP-hard for the parameter  $d$ . More precisely, it is  $\mathbb{W}[1]$ -hard for parameter  $\Delta$ , the degree of  $\mathcal{T}$ , even when  $d = 1$ .

**Proof** As illustrated in Fig. 5, this can be shown by reduction from MULTI-COLORED CLIQUE (Given a graph  $G$ , an integer  $k$  and a partition of the vertices of  $G$  into  $k$  sets, is there a clique in  $G$  containing exactly one vertex from each of the  $k$  sets?). Consider a  $k$ -partite graph  $G = (V, E)$  with  $V = \bigcup_{i=1}^k V_i$ . We assume that  $G$  is regular (each vertex has degree  $\delta$  and that each  $V_i$  has the same size  $n$  (MULTI COLORED CLIQUE is  $\mathbb{W}[1]$ -hard under these restrictions [28, 29]). Let  $L := \delta k - \binom{k}{2}$  and  $N = \max\{|V|, L + 1\}$ . We now build a graph  $G'$  and a tree-decomposition  $\mathcal{T}'$ : start with  $G' := G$ . Add  $k$  independent cliques  $K_1, \dots, K_k$  of size  $N + 1$ . Add  $k$  sets of  $N$  vertices  $Z_i$  ( $i \in [k]$ ) and, for each  $i \in [k]$ , add edges between each  $v \in V_i$  and each  $z \in Z_i$ . Build  $\mathcal{T}$  using  $2k + 1$  bags  $T_0, T_{1,i}, T_{2,i}$  for  $i \in [k]$ , such that  $T_0 = V$ ,  $T_{1,i} = V_i \cup K_i$  and  $T_{2,i} = V_i \cup Z_i$ . The tree-decomposition is completed by connecting  $T_{2,i}$  to  $T_{1,i}$  and  $T_{1,i}$  to  $T_0$



**Fig. 5** Reduction for Theorem 4 showing that TREE-DIET is NP-hard even for  $d = 1$ , from a graph  $G$  (left) with  $k = 3$  and  $n = 3$  to a graph  $G'$  (right), given by its tree-decomposition of width  $N + n + 1$ : a 1-tree-diet for  $G'$  amounts to selecting a  $k$ -clique in the root bag, i.e. in  $G$

for each  $i \in [k]$ . Thus,  $\mathcal{T}$  is a tree-decomposition of  $G'$  with  $\Delta = k$  and maximum bag size  $n + N + 1$  (vertices of  $V$  induce a size-3 path in  $\mathcal{T}$ , other vertices appear in a single bag, edges of  $G$  appear in  $T_0$ , edges of  $K_i$  in  $T_{1,i}$ , and finally edges between  $V_i$  and  $Z_i$  appear in  $T_{2,i}$ ). The following claim completes the reduction:

$\mathcal{T}$  has a 1-tree-diet losing at most  $L$  edges from  $G' \Leftrightarrow G$  has a  $k -$  clique.

$\Leftarrow$  Assume  $G$  has a  $k$ -clique  $X = \{x_1, \dots, x_k\}$  (with  $x_i \in V_i$ ). Build  $\mathcal{T}'$  by removing each  $x_i$  from bags  $T_0$  and  $T_{1,i}$ . Then  $\mathcal{T}'$  is a 1-tree-diet of  $\mathcal{T}$ . There are no edges lost by removing  $x_i$  from  $T_{1,i}$  (since  $x_i$  is not connected to  $K_i$ ), and the edges lost in  $T_0$  are all edges of  $G$  adjacent to any  $x_i$ . Since  $X$  forms a clique and each  $x_i$  has degree  $\delta$ , there are  $L = k\delta - \binom{k}{2}$  such edges.

$\Rightarrow$  Consider a 1-tree-diet  $\mathcal{T}'$  of  $\mathcal{T}$  losing  $L$  edges. Since each bag  $T_{1,i}$  has maximum size,  $\mathcal{T}'$  must remove at least one vertex  $x_i$  in each  $T_{1,i}$ . Note that  $x_i \in V_i$  (since removing  $x_i \in K_i$  would loose at least  $N \geq L + 1$  edges). Furthermore,  $x_i$  may not be removed from  $T_{2,i}$  (otherwise  $N$  edges between  $x_i$  and  $Z_i$  would be lost), so  $x_i$  must also be removed from  $T_0$ . Let  $K$  be the number of edges in  $G[\{x_1 \dots x_k\}]$ . The total number of lost edges in  $T_0$  is  $\delta k - K$ . Thus, we have  $\delta k - K \leq L$  and  $K \geq \binom{k}{2}$ :  $\{x_1, \dots, x_k\}$  form a  $k$ -clique of  $G$ .  $\square$

**Theorem 5** *PATH-DIET is  $\mathbb{W}[1]$ -hard for parameter  $d$ .*

**Proof** By reduction from CLIQUE. Given a  $\delta$ -regular graph  $G$  with  $n$  vertices and  $m$  edges and an integer  $k$ , consider the trivial tree-decomposition  $\mathcal{T}$  of  $G$  with a single bag containing all vertices of  $G$  (it has width  $n - 1$ ). Then  $(\mathcal{T}, G)$  has a  $k$ -tree-diet losing  $\delta k - \binom{k}{2}$  edges if and only if  $G$  has a  $k$ -clique. Indeed, such a tree-diet  $\mathcal{T}'$  would remove a set  $X$  of  $k$  vertices from  $G$  and losing  $\delta k - \binom{k}{2}$  edges, so  $X$  induces  $\binom{k}{2}$  edges and is a  $k$ -clique of  $G$ . Any instance  $G$ , with parameter  $k$ , of CLIQUE can therefore be transformed into an equivalent instance  $(\mathcal{T}, G)$  of PATH-DIET, with parameter  $d = k$ . Since it qualifies as a parameterized reduction, PATH-DIET is  $\mathbb{W}[1]$ -hard.  $\square$

**FPT algorithm**

**For general tree-decompositions**

We describe here a  $O(3^{tw}n)$ -space,  $O(\Delta^{tw+2} \cdot 6^{tw}n)$ -time dynamic programming algorithm for the TREE-DIET problem, with  $\Delta$  and  $tw$  being respectively the maximum number of children of a bag in the input tree-decomposition and its width. On *binary* tree-decompositions

(where each bag has at most 2 children), it yields a  $O(3^{tw}n)$ -space  $O(12^{tw}n)$ -time FPT algorithm.

**Coloring formulation**

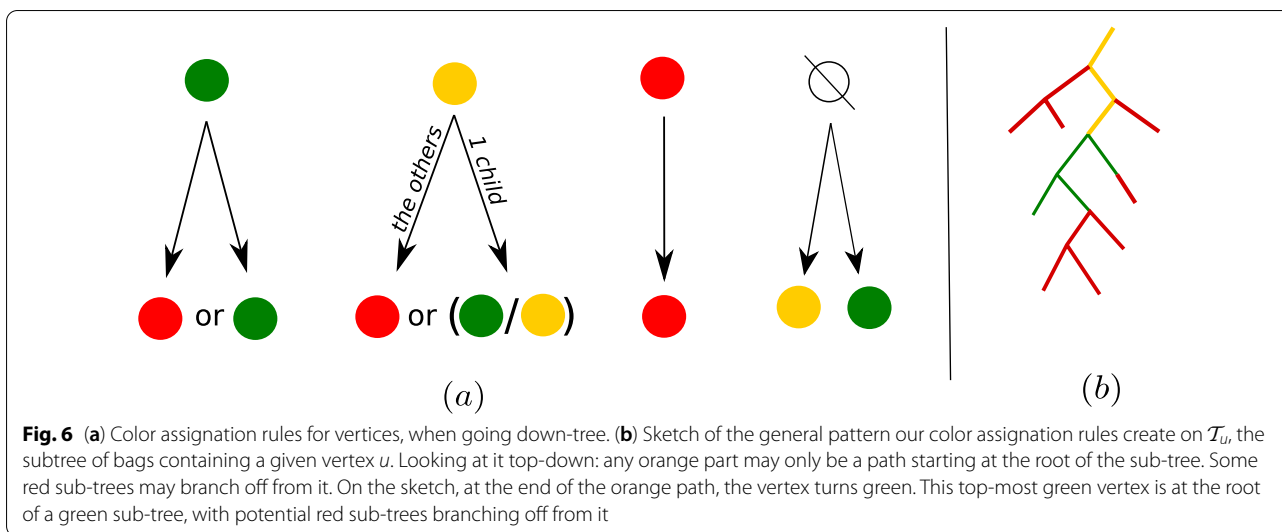
We aim at solving the following problem: given a tree-decomposition  $\mathcal{T}$  of width  $tw$  of a graph  $G$ , we want to remove vertices from the bags of  $\mathcal{T}$  to reach a target width  $tw'$  while *losing* as few edges from  $G$  as possible. We tackle the problem through an equivalent *coloring* formulation: our algorithm will assign a color to each occurrence of a vertex in the tree decomposition. We work with three colors: red ( $\color{red}{\text{r}}$ ), orange ( $\color{orange}{\text{o}}$ ), and green ( $\color{green}{\text{g}}$ ). Green means that the vertex is kept in the bag, while orange and red means removal of the vertex. An edge is thus visible within a bag when both its ends are green. It is lost if there is no bag where it is visible. To ensure equivalence with the original problem, the colors will be assigned following local rules, which we now describe.

**Definition 1** A coloring of vertices in the bags of the decomposition is said to be *valid* if it follows the following rules:

- A vertex of a bag not present in its parent may be green or orange (R1)
- A green vertex in a bag may be either green or red in its children (R2)
- A red vertex in a bag must stay red in its children (R3)
- An orange vertex in a bag has to be either orange or green in exactly one child (unless there is no child with this vertex), and must be red in the other children (R4)

These rules are summarized in Fig. 6a.

When going down the tree, a green vertex may only stay green or permanently become red. As for orange vertices, they are locally absent but “may potentially be found further down the tree”, while red vertices are removed from both the current bag and its entire subtree. An immediate consequence of these rules is therefore that the green occurrences of a given vertex form a (possibly empty) connected subtree. (R4) in particular is crucial to this connectivity: if an orange vertex could become orange in several children, it would be able to turn green in several disconnected subtrees. Figure 6b shows an example sketch for a valid coloring of the occurrences of a given vertex in the tree-decomposition. A vertex may only be orange along a path starting from its highest occurrence in the tree, with any part branching off that path entirely red. It ends at the top of a (potentially



**Fig. 6** (a) Color assignation rules for vertices, when going down-tree. (b) Sketch of the general pattern our color assignation rules create on  $\mathcal{T}_u$ , the subtree of bags containing a given vertex  $u$ . Looking at it top-down: any orange part may only be a path starting at the root of the sub-tree. Some red sub-trees may branch off from it. On the sketch, at the end of the orange path, the vertex turns green. This top-most green vertex is at the root of a green sub-tree, with potential red sub-trees branching off from it

empty) green subtree, whose vertices may also be parents to entirely red subtrees.

We will now more formally prove the equivalence of the coloring formulation to the original problem. Let us first introduce two definitions. Given a valid coloring  $\mathcal{C}$  of a tree-decomposition of  $G$ , an edge  $(u, v)$  of  $G$  is said to be *realizable* if there exists a bag in which both  $u$  and  $v$  are green per  $\mathcal{C}$ . Given an integer  $d$ , a coloring  $\mathcal{C}$  of  $\mathcal{T}$  is said to be  $d$ -*diet-valid* if removing red/orange vertices reduces the width of  $\mathcal{T}$  from  $w(\mathcal{T})$  to  $w(\mathcal{T}) - d$ .

**Proposition 1** *Given a graph  $G$ , a tree-decomposition  $\mathcal{T}$  of width  $tw$ , and a target width  $tw' < tw$ , The TREE-DIET problem is equivalent to finding a  $(tw - tw')$ -diet-valid coloring  $\mathcal{C}$  of  $\mathcal{T}$  allowing for a number of realizable edges in  $G$  as large as possible.*

**Proof** Given a  $(tw - tw')$ -tree-diet of  $\mathcal{T}$  specifying which vertices are removed from which bags, we first show how to obtain a valid coloring  $\mathcal{C}$  for  $\mathcal{T}$  incurring the same number of lost (unrealizable) edges. Let us denote by  $\mathcal{T}'$  the tree decomposition of width  $tw'$  obtained by applying the diet to  $\mathcal{T}$ . To start with, a vertex  $u$  is colored green in the bags where it is not removed. By the validity of  $\mathcal{T}'$  as a decomposition, this set of bags forms a connected subtree, that we denote  $\mathcal{T}_u^g$ . We also write  $\mathcal{T}_u$  for the subtree of bags containing  $u$  in the original decomposition  $\mathcal{T}$ . If  $\mathcal{T}_u^g$  and  $\mathcal{T}_u$  do not have the same root, then  $u$  is colored orange on the the path in  $\mathcal{T}$  from the root of

$\mathcal{T}_u$  (included) and the root of  $\mathcal{T}_u^g$  (excluded). Vertex  $u$  is colored red in any other bag of  $\mathcal{T}_u$  not covered by these two cases. The resulting coloring follows rules (R1-4) and induces the same set of lost/non-realizable edges as the original  $(tw - tw')$ -tree-diet. Conversely, an equivalent  $(tw - tw')$ -tree-diet is obtained from a  $(tw - tw')$ -diet-valid coloring by removing red/orange vertices and keeping green ones. If a given vertex has no green occurrences, it is entirely removed from the tree decomposition and all its edges are *lost* (it becomes an isolated vertex). We may add it back to the tree decomposition by introducing a new bag containing only this vertex, which we connect arbitrarily to the tree decomposition.  $\square$

**Decomposition of the search space and sub-problems**

Based on this coloring formulation, we now describe a dynamic programming scheme for the TREE-DIET problem. We work with sub-problems indexed by tuples  $(X_i, f)$ , with  $X_i$  a bag of the input tree decomposition and  $f$  a coloring of the vertices of  $X_i$  in green, orange or red (in particular,  $f^{-1}(g)$  denotes the green vertices of  $X_i$ , and similarly for  $o$  and  $r$ ).

Let us introduce some notations before giving the definition of our dynamic programming table. Given an edge  $(u, v)$  of  $G$ , realizable when coloring a tree-decomposition  $\mathcal{T}$  of  $G$  with  $\mathcal{C}$ , we write  $\mathcal{T}_{uv}^g$  the subtree of  $\mathcal{T}$  in which



both  $u$  and  $v$  are green. We denote by  $\mathcal{T}_i$  the subtree of the decomposition rooted at  $X_i$ , and  $C(i, f)$  the  $d$ -diet-valid colorings of  $\mathcal{T}_i$  agreeing with  $f$  on  $i$ , with  $d = tw - tw'$ . Our dynamic programming table is then defined as:

$$c(X_i, f) = \begin{cases} \max_{C \in \mathcal{C}(i, f)} & \left| \left\{ \begin{array}{l} \text{Edges } (u, v), \text{ realizable within } \mathcal{T}_i \text{ colored with } C \\ \text{such that } \mathcal{T}_{uv}^g \text{ is entirely contained strictly below } X_i \end{array} \right\} \right| \\ -\infty & \text{if } f \text{ assigns green to at most } tw' + 1 \text{ vertices} \\ & \text{otherwise.} \end{cases}$$

The cell  $c(X_i, f)$  therefore aggregates all edges realizable strictly below  $X_i$ . As we shall see through the recurrence relation below and its proof, edges with both ends green in  $X_i$  will be accounted for above  $X_i$  in  $\mathcal{T}$ .

We assume w.l.o.g that the tree-decomposition is rooted at an empty bag  $R$ . Given the definition of the table, the maximum number of realizable edges, compatible with a tree-diet of  $(tw - tw')$  to  $\mathcal{T}$ , can be found in  $c(R, \emptyset)$ .

The following theorem presents a recurrence relation obeyed by  $c(X_i, f)$ :

**Theorem 6** For a bag  $X_i$  of  $\mathcal{T}$ , with children  $Y_1, \dots, Y_\Delta$ , we have:

$$c(X_i, f) = \max_{m: f^{-1}(\circ) \rightarrow [1.. \Delta]} \left[ \sum_{1 \leq j \leq \Delta} \left( \max_{f'_j \in \text{compatible}(Y_j, f, m)} c(Y_j, f'_j) + |count(f, f'_j)| \right) \right],$$

with

- $m$ : a map from the orange vertices in  $X_i$  to the children of  $X_i$ . It decides for each orange vertex  $u$ , which child, among those which contain  $u$ , will color  $u$  orange or green; If there are no orange vertices in  $X_i$ , only the trivial empty map is considered.
- $\text{compatible}(Y_j, f, m)$ : the set of colorings of  $Y_j$  compatible with  $f$  on  $X_i$  and  $m$ ;
- $count(f, f'_j)$ : set of edges of  $G$  involving two vertices of  $Y_j$  green by  $f'_j$ , but such that one of them is either not in  $X_i$  or not green by  $f$ .

Note that  $\text{compatible}(Y_j, f, m)$  may contain colorings  $f'_j$  that colour too many vertices in  $Y_j$  in green to reach target width  $tw'$ . In that case  $c(Y_j, f'_j) = -\infty$ .

Theorem 6 relies on the following separation lemma for realizable edges under a valid coloring of a tree-decomposition. Recall that we suppose w.l.o.g that the tree-decomposition is rooted at an empty bag.

**Lemma 1** An edge  $(u, v)$  of  $G$ , realizable in  $\mathcal{T}$  under  $\mathcal{C}$ , is contained in exactly one set of the form  $count(C_{|P}, C_{|X})$  with  $X$  a bag of  $\mathcal{T}$  and  $P$  its parent,  $C_{|P}, C_{|X}$  the restrictions of  $\mathcal{C}$  to  $P$  and  $X$ , respectively, and “count” defined as above.

In addition,  $X$  is the root of the subtree of  $\mathcal{T}$  in which both  $u$  and  $v$  are green.

**Proof** Given, in a tree-decomposition, a bag  $P$  colored with  $f$ , with a child  $X$  colored with  $h$ , a more precise definition for  $count(f, h)$  is:

$$count(f, h) = \left\{ (u, v) \in E \mid \begin{array}{l} h(u) = h(v) = g \text{ and} \\ (u \notin P \text{ or } f(u) \neq g \text{ or } v \notin P \text{ or } f(v) \neq g) \end{array} \right\}.$$

Now, given a realizable edge  $(u, v)$ , in a tree-decomposition  $\mathcal{T}$  colored with  $\mathcal{C}$ , the set of bags in which both  $u$  and  $v$  are green forms a connected subtree of  $\mathcal{T}$ . This subtree has a root, or lowest common ancestor, that we denote  $R_{(u,v)}$ . Since we assumed  $\mathcal{T}$  to be rooted at an empty bag,

$R_{(u,v)}$  is not the root of  $\mathcal{T}$ , and has a parent. We call this parent  $P_{(u,v)}$ . Clearly,  $(u, v)$  belongs to the “count set” associated to the edge  $(P_{(u,v)}) \rightarrow (R_{(u,v)})$  of  $\mathcal{T}$ , while for any other edge  $X \rightarrow Y$  of  $\mathcal{T}$ , the colors of  $u$  and  $v$  cannot verify the conditions to belong to the associated “count set”. □

**Proof of Theorem 6** [≤] Let us more concisely use  $RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)$  to denote the set of edges  $(u, v)$  of  $G$ , realizable under the  $(tw - tw')$ -diet-valid coloring  $\mathcal{C}$  of  $\mathcal{T}_i$ , such that  $\mathcal{T}_{uv}^g$  is entirely contained strictly below  $X_i$ . We have, if  $f$  contains enough red/orange vertices to reduce the size of  $X_i$  to target size:

$$c(X_i, f) = \max_{C \in \mathcal{C}(i, f)} |RE_{\downarrow}(\mathcal{T}_i, C, G)|.$$

By definition,  $c(X_i, f)$  is the maximum number of realizable edges in the subtree-decomposition rooted at  $X_i$ , such that all green-green occurrences

of the edge occur strictly below  $X_i$ , and under the constraint that  $f$  colors  $X_i$ . Let  $\mathcal{C}$  be a coloring for  $\mathcal{T}_i$  realizing the optimum  $c(X_i, f)$ . Its restrictions to  $Y_1 \dots Y_\Delta$  yield colorings  $f'_1 \dots f'_\Delta$ . Likewise, its restrictions to the subtree-decompositions  $\mathcal{T}'_1 \dots \mathcal{T}'_\Delta$  rooted at  $Y_1 \dots Y_\Delta$  yield colorings  $\mathcal{C}'_1 \dots \mathcal{C}'_\Delta$  compatible with  $f'_1 \dots f'_\Delta$ .  $\mathcal{C}'_1 \dots \mathcal{C}'_\Delta$  cannot be better than the optimal, so  $\forall j, |RE_\downarrow(\mathcal{T}'_j, \mathcal{C}'_j, G)| \leq c(Y_j, f'_j)$ . Let  $(u, v)$  be an edge of  $RE_\downarrow(\mathcal{T}_i, \mathcal{C}, G)$ . Per Lemma 1, either  $(u, v) \in \text{count}(f, f'_j)$  for some  $j$  (if  $Y_j$  is the root of  $\mathcal{T}_{uv}^g$ ) and  $(u, v) \notin \cup_j RE_\downarrow(\mathcal{T}'_j, \mathcal{C}'_j, G)$  or  $(u, v) \in \text{count}(f, f'_j)$  and  $\exists j$  such that  $(u, v) \in RE_\downarrow(\mathcal{T}'_j, \mathcal{C}'_j, G)$ . Therefore:

$$c(X_i, f) = |RE_\downarrow(\mathcal{T}_i, \mathcal{C}, G)| = \sum_{1 \leq j \leq \Delta} \left[ |RE_\downarrow(\mathcal{T}'_j, \mathcal{C}'_j, G)| + \text{count}(f, f'_j) \right] \leq \sum_{1 \leq j \leq \Delta} \left( c(Y_j, f'_j) + \text{count}(f, f'_j) \right),$$

and, a fortiori

$$c(X_i, f) \leq \max_{m: f^{-1}(\circ) \rightarrow [1 \dots \Delta]} \sum_{1 \leq j \leq \Delta} \max_{f'_j \in \text{compatible}(Y_j, f, m)} \left( c(Y_j, f'_j) + \text{count}(f, f'_j) \right).$$

$\square$  Conversely, given  $f$ , let  $m$  be an assignation map for orange vertices and  $f'_1 \dots f'_\Delta$  colorings of  $Y_1 \dots Y_\Delta$  compatible with  $f$  and  $m$ , and let  $\mathcal{C}'_1 \dots \mathcal{C}'_\Delta$  be colorings of  $\mathcal{T}'_1 \dots \mathcal{T}'_\Delta$  realizing the optima  $c(Y_1, f'_1) \dots c(Y_\Delta, f'_\Delta)$ . The union of  $\mathcal{C}'_1 \dots \mathcal{C}'_\Delta$  and  $f$  is a coloring  $\mathcal{C}$  for  $\mathcal{T}_i$ , the subtree-decomposition rooted at  $X_i$ , which can not be better than optimal ( $|RE_\downarrow(\mathcal{T}_i, \mathcal{C}, G)| \leq c(X_i, f)$ ). As before, an edge  $(u, v)$  either belongs to  $\cup_j \text{count}(f, f'_j)$  or to  $\cup_j RE_\downarrow(\mathcal{T}'_j, \mathcal{C}'_j, G)$  but not both. In any case, it belongs to  $RE_\downarrow(\mathcal{T}_i, \mathcal{C}, G)$ . Therefore:

$$\sum_{1 \leq j \leq \Delta} \left( c(Y_j, f'_j) + \text{count}(f, f'_j) \right) = \sum_{1 \leq j \leq \Delta} \left( |RE_\downarrow(\mathcal{T}'_j, \mathcal{C}'_j, G)| + \text{count}(f, f'_j) \right) = |RE_\downarrow(\mathcal{T}_i, \mathcal{C}, G)| \leq c(X_i, f).$$

This is true for any choice of  $m, f'_1 \dots f'_\Delta$ , therefore:

$$\max_{m: f^{-1}(\circ) \rightarrow [1 \dots \Delta]} \sum_{1 \leq j \leq \Delta} \max_{f'_j \in \text{compatible}(Y_j, f, m)} \left( c(Y_j, f'_j) + \text{count}(f, f'_j) \right) \leq c(X_i, f),$$

which concludes the proof.  $\square$

*Dynamic programming algorithm* The recurrence relation of Theorem 6 naturally yields a dynamic

programming algorithm for the TREE-DIET problem, as stated below:

**Theorem 7** *There exists a  $O(\Delta^{tw+2} \cdot 6^{tw} \cdot n)$ -time,  $O(3^{tw} \cdot n)$ -space algorithm for the TREE-DIET problem, with  $\Delta$  the maximum number of children of a bag in the input tree-decomposition, and  $tw$  its width.*

**Proof (Proof of Theorem 7)** Given the coloring formulation and Proposition 1, and given the sub-problems and  $c(X_i, f)$ -table definitions, with  $R$  the (empty) root of

the tree-decomposition,  $c(R, \emptyset)$  is indeed the maximum possible number of realizable edges when imposing a  $(tw - tw')$ -diet to  $\mathcal{T}$ . The recurrence relation of Theorem 6 therefore lends itself to a dynamic programming approach, over the tree-decomposition  $\mathcal{T}$  following leaf-to-root order, for the problem.

It is reasonable to assume the number of bags in a tree decomposition to be linear in  $n$  (this is for instance the case for a *nice* tree decomposition [29, 42], or for a tree decomposition obtained from an *elimination ordering*, see [17, 43]). Therefore, the number of entries to the table is  $O(3^{tw}n)$ , given that a bag  $X$  may be colored in  $3^{|X|}$  ways, and that the maximum size of  $X$  is  $tw + 1$ . For a given entry  $X_i$ , one must first enumerate all possible choices of  $m : f^{-1}(\circ) \rightarrow [1 \dots \Delta]$ , map assigning one child of  $X_i$  to each orange vertex in  $X_i$ . There are  $O(\Delta^{tw+1})$  possibilities for  $m$  in the worst case, as  $|f^{-1}(\circ)| \leq tw + 1$ . Then, for each child  $Y_j$ , one must enumerate all possible colorings  $f'_j$  compatible with  $f$ . Possibilities for  $f'_j(u)$  depend on the color by  $f$ :

- if  $u \notin X_i \rightarrow f'_j(u) = \circ$  or  $g$
- if  $f(u) = g \rightarrow f'_j(u) = g$  or  $r$
- if  $f(u) = \circ \rightarrow f'_j(u) = \circ$  or  $g$  if  $m[u] = j$  or  $f'_j(u) = r$  otherwise.

- if  $f(u) = r \rightarrow f'_j(u) = r$

Overall, as there are at most  $\Delta$  children,  $tw + 1$  vertices in each child, and 2 possibilities (see enumeration of cases above) of color for each vertex in a child, yielding a total number of compatible colorings bounded by  $O(\Delta \cdot 2^{tw+1})$ . Multiplying these contributions, the overall time complexity of our algorithm is therefore  $O(\Delta^{tw+2} \cdot 6^{tw} \cdot n)$ .  $\square$

**Corollary 1** *BINARY-TREE-DIET* ( $\Delta = 2$ ) admits an FPT algorithm for the  $tw$  parameter.

A pseudo-code implementation of the algorithm, using memoization, is included in Additional file 1: Section B

**For path decompositions**

In the context of path decompositions, we note that the number of removed vertices per bag can be limited to at most  $2d$  without losing the optimality. More precisely, we say that a coloring  $C$  is  $d$ -simple if any bag has at most  $d$  orange and  $d$  red vertices. We obtain the following result, using transformations given in Fig. 7.

**Proposition 2** Given a graph  $G$  and a path-decomposition  $T$ , if  $C$  is a  $d$ -diet-valid coloring of  $T$  losing  $k$  edges, then  $T$  has a  $d$ -diet-valid coloring that is  $d$ -simple, and loses at most  $k$  edges.

**Proof of Proposition 2** Consider such a coloring  $C$  with a maximal number of green vertices. We show that it is  $d$ -simple. Assume the path-decomposition  $T$  is rooted in bag  $X_1$  and each  $X_i$  is the parent of  $X_{i+1}$ . Pick  $i$  to be the smallest index so that at least  $d + 1$  vertices in  $X_i$  are

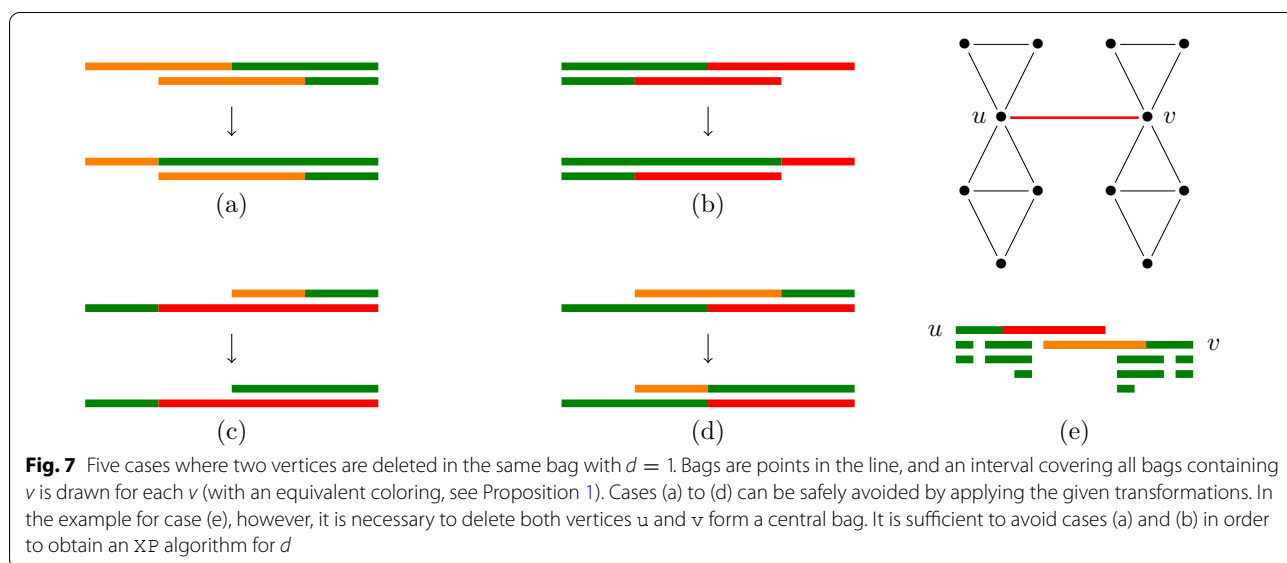
colored red by  $C$ , assume any such  $i$  exists. Then one of these vertices, say  $u$ , is not colored red in  $X_{i-1}$  (either because  $i = 1$ , or it is not in  $X_{i-1}$ , or it is orange or green in  $X_{i-1}$ ). Consider  $C'$  obtained by  $C$  and coloring  $u$  green in  $X_i$ . Then  $C'$  satisfies local rules R1 through R4 (a green vertex may be absent, green or orange in the parent bag, and a red vertex may be green in the parent bag). Furthermore, it is  $d$ -diet-valid since it still removes at least  $d$  (red) vertices in  $X_i$ . Overall  $C'$  is another  $d$ -diet-valid coloring with more green vertices: a contradiction, so no such  $i$  exist (and no bag has  $d + 1$  red vertices). The same argument works symmetrically for orange vertices. Overall,  $C$  is  $d$ -simple.  $\square$

Together with Proposition 1, this shows that it is sufficient to restrict our algorithm to  $d$ -simple colorings. (See also Fig. 7). In particular, for any set  $X_i$ , choosing which  $\leq d$  vertices are orange and which  $\leq d$  are red, among the total of  $n$  vertices, is enough to fix a coloring. The number of such colorings is therefore bounded by  $O(tw^{2d})$ . Applying this remark to our algorithm presented in Sect. 4.1 yields the following result:

**Theorem 8** *PATH-DIET* can be solved in  $O(tw^{2d}n)$ -space and  $O(tw^{4d}n)$ -time.

**Proofs of concept**

We now illustrate the relevance of our approach, and the practicality of our algorithm for TREE-DIET, by using it in conjunction with FPT algorithms for three problems in RNA bioinformatics. We implemented in C++ the dynamic programming scheme described in Theorem 7



and Additional file 1: Section B. Its main primitives are made available for Python scripting through pybind11 [44].

It actually allows to solve a generalized *weighted* version of TREE DIET, as explained in Additional file 1: Section B. This feature allows to favour the conservation of important edges (e.g. RNA backbone) during simplification, by assigning them a much larger weight compared to other edges. Our implementation is freely available at <https://gitlab.inria.fr/amibio/tree-diet>.

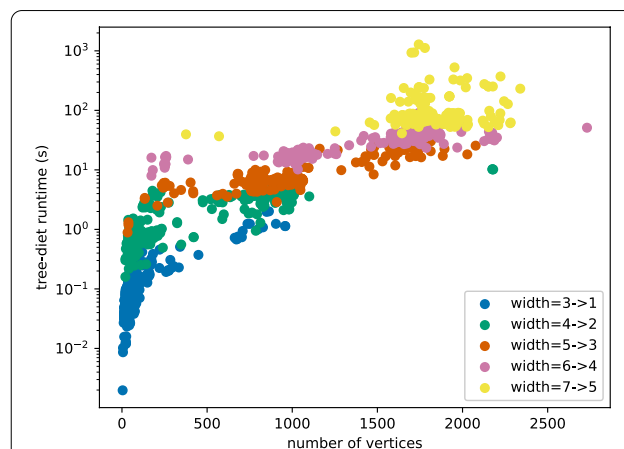
The execution time of this implementation on elements of the data set used for Fig. 1 (all RNA-only structures of the PDB database) is represented on Figure 8, for input treewidth values of up to 7. It shows that our tree-diet method is applicable with reasonable run-times ( $\lesssim 1$  h) for all structures of width  $\leq 7$ . The proofs-of-concepts presented in this section involve however instances with treewidth of up to 9, in the case of RNA design, for which the run-time also stays reasonable.

### Memory-parsimonious unbiased sampling of RNA designs

As a first use case for our simplification algorithm, we strive to ease the sampling phase of a recent method, called RNAPond [22], addressing RNA negative design. The method targets a set of base pairs  $S$ , representing a secondary structure of length  $n$ , and infers a set  $\mathcal{D}$  of  $m$  disruptive base pairs (DBPs) that must be avoided. It relies on a  $\Theta(k \cdot (n + m))$  time algorithm for sampling  $k$  random sequences (see Additional file 1: Section C for details) after a preprocessing in  $\Theta(n \cdot m \cdot 4^{tw})$  time and  $\Theta(n \cdot 4^{tw})$  space. Here, the input consists of a graph  $G = ([1, n], S \cup \mathcal{D})$  and a tree decomposition  $\mathcal{T}$  of  $G$ , having width  $tw$ . In practice, the preprocessing largely dominates the overall runtime, even for large values of  $k$ , and its large memory consumption represents the main bottleneck.

This discrepancy in the complexities/runtimes of the preprocessing and sampling suggests an alternative strategy: relaxing the set of constraints to  $(S', \mathcal{D}')$ , with  $(S' \cup \mathcal{D}') \subset (S \cup \mathcal{D})$ , and compensating it through a rejection of sequences violating constraints in  $(S, \mathcal{D}) \setminus (S', \mathcal{D}')$ . The relaxed algorithm would remain unbiased, while the average-case time complexity of the rejection algorithm would be in  $\Theta(k \cdot \bar{q} \cdot (n + m))$  time, where  $\bar{q}$  represents the relative increase of the partition function ( $\approx$  the sequence space) induced by the relaxation. The preprocessing step would retain the same complexity, but based on a (reduced) treewidth  $tw' \leq tw$  for the relaxed graph  $G' = ([1, n], S' \cup \mathcal{D}')$ .

These complexities enable a tradeoff between the rejection (time), and the preprocessing (space), which may be critical to unlock future applications of RNA design. Indeed, the treewidth can be decreased by removing



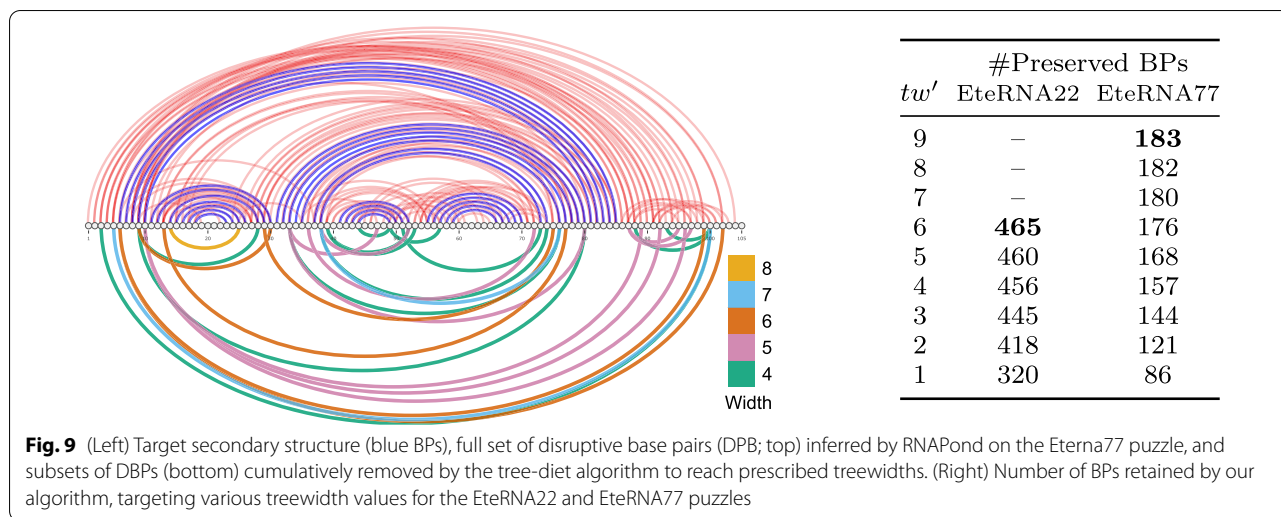
**Fig. 8** Run-time of the tree-diet algorithm on all RNA-only structures of the PDB database, versus the size (length of the RNA string) of these structures. The data set is the same as Fig. 1, limited to structures of treewidth  $\leq 7$ . Structures are colored by their original treewidth. Here, we have asked the algorithm to reduce the treewidth by 2

relatively few base pairs, as demonstrated below using our algorithm on pairs inferred for hard design instances.

We considered sets of DBPs inferred by RNAPond over two puzzles in the EteRNA benchmark. The EteRNA22 puzzle is an empty secondary structure spanning 400 nts, for which RNAPond obtains a valid design after inferring 465 DBPs. A tree decomposition of the graph formed by these 465 DBPs is then obtained with the standard min-fill-ordering heuristic [18], giving a width of 6. The EteRNA77 puzzle is 105 nts long, and consists in a collection of helices interspersed with destabilizing internal loops. RNAPond failed to produce a solution, and its final set of DBPs consists of 183 pairs, for which the same heuristic yields a tree decomposition of width 9. We further make both tree decompositions binary through bag duplications (see Supplementary Section A), giving an FPT runtime to our algorithm, while potentially lowering the number of lost edges.

Executing the tree-diet algorithm (Theorem 7) on both graphs and their tree decompositions, we obtained simplified graphs, having lower treewidth while typically losing few edges, as illustrated and reported in Fig. 9. Remarkably, the treewidth of the DBPs inferred for EteRNA22 can be decreased to  $tw' = 5$  by only removing 5 DBPs/edges (460/465 retained), and to  $tw' = 4$  by removing 4 further DBPs (456/465). For EteRNA77, our algorithm reduces the treewidth from 9 to 6 by only removing 7 DBPs.

Rough estimates can be provided for the tradeoff between the rejection and preprocessing complexities, by assuming that removing a DBP homogeneously increases the value of  $\mathcal{Z}$  by a factor  $\alpha := 16/10$  (#pairs/#incomp.



pairs). The relative increase in partition function is then  $\bar{q} \approx \alpha^b$ , when  $b$  base pairs are removed. For EteRNA22, reducing the treewidth by 2 units (6→4), i.e. a 16 fold reduction of the memory and preprocessing time, can be achieved by removing 9 DBPs, i.e. a 69 fold expected increase in the time of the generation phase. For EteRNA77, the same 16 fold ( $tw' = 9 \rightarrow 7$ ) reduction of the preprocessing time/space can be achieved through an estimated 4 fold increase of the generation time. A more aggressive 256 fold memory gain can be achieved at the expense of an estimated 1 152 fold increase in generation time. Given the large typical asymmetry in runtimes and implementation constants between the computation-heavy preprocessing and, relatively light, generation phases, the availability of an algorithm for the TREE-DIET problem provides new options, especially to circumvent memory limitations.

### Structural alignment of complex RNAs

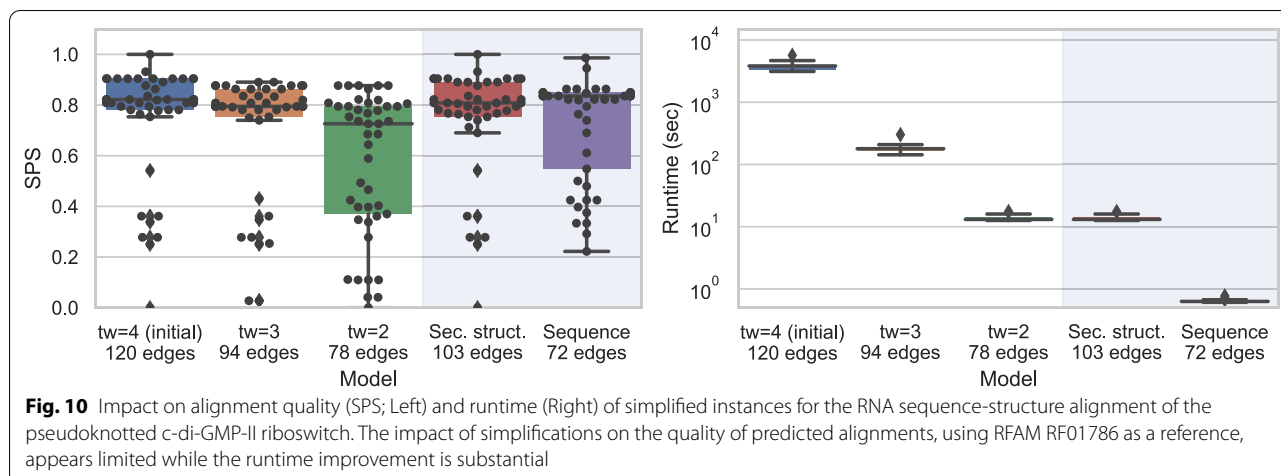
Structural homology is often posited within functional families of non-coding RNAs, and is foundational to algorithmic methods for multiple RNA alignments [13], considering RNA base pairs while aligning distant homologs. In the presence of complex structural features (pseudo-knots, base triplets), the sequence-structure alignment problem becomes hard, yet admits XP solutions based on the treewidth of the base pair + backbone graph. In particular, Rinaudo et al. [12] describe a  $\Theta(n.m^{tw+1})$  algorithm for optimally aligning a structured RNA of length  $n$  onto a genomic region of length  $m$ . It optimizes an alignment score that includes: (i) substitution costs for matches/mismatches of individual nucleotides and base pairs (including arc-breaking) based on the RIBOSUM

matrices [45]; and (ii) an affine gap cost model [46]. We used the implementation of the Rinaudo et al. algorithm, implemented in the LicoRNA software package [47, 48].

### Impact of treewidth on the structural alignment of a riboswitch

In this case study, we used our tree-diet algorithm to modulate the treewidth of complex RNA structures, and investigate the effect of the simplification on the quality and runtimes of structure-sequence alignments. We considered the Cyclic di-GMP-II riboswitch, a regulatory motif found in bacteria that is involved in signal transduction, and undergoes conformational change upon binding the second messenger c-di-GMP-II [49, 50]. A 2.5Å resolution 3D model of the c-di-GMP-II riboswitch in *C. acetobutylicum*, proposed by Smith et al. [51] based on X-ray crystallography, was retrieved from the PDB [24] (PDBID: 3Q3Z). We annotated its base pairs geometrically using the DSSR method [52]. The canonical base pairs, supplemented with the backbone connections, were then accumulated in a graph, for which we heuristically computed an initial tree decomposition  $\mathcal{T}_4$ , having treewidth  $tw = 4$ .

We simplified our the initial tree decomposition  $\mathcal{T}_4$ , and obtained simplified models  $\mathcal{T}_3$ , and  $\mathcal{T}_2$ , having width  $tw' = 3$  and 2 respectively. As controls, we included tree decompositions based on the secondary structure (max. non-crossing set of BPs;  $\mathcal{T}_{2D}$ ) and sequence ( $\mathcal{T}_{1D}$ ). We used LicoRNA to predict an alignment  $a_{\mathcal{T},w}$  of each original/simplified tree decomposition  $\mathcal{T}$  onto each sequence  $w$  of the c-di-GMP-II riboswitch family in the RFAM database [13] (RF01786). Finally, we reported the LicoRNA runtime, and computed the Sum of Pairs Score



(SPS) [53] as a measure of the accuracy of  $a_{T,w}$  against a reference alignment  $a_w^*$ :

$$SPS(a_{T,w}; a_w^*) = \frac{|\text{MatchCols}(a_{T,w}) \cap \text{MatchCols}(a_w^*)|}{|\text{MatchCols}(a_w^*)|},$$

the alignment  $a_w^*$  between the 3Q3Z sequence and  $w$  induced by the manually-curated RFAM alignment of the RF01786 family.

The results, presented in Fig. 10, show a limited impact of the simplification on the quality of the predicted alignment, as measured by the SPS in comparison with the RFAM alignment. The best average SPS (77.3%) is achieved by the initial model, having treewidth of 4, but the average difference with simplified models appears very limited (e.g. 76.5% for  $T_3$ ), especially when considering the median. Meanwhile, the runtimes mainly depend on the treewidth, ranging from 1h for  $T_4$  to 300ms for  $T_{1D}$ . Overall,  $T_{2D}$  seems to represent the best compromise between runtime and SPS, although its SPS may be artificially inflated by our election of RF01786 as our reference (built from a covariance model, i.e. essentially a 2D structure). Finally, the difference in number of edges (and induced SPS) between  $T_{2D}$  and  $T_2$ , both having  $tw = 2$ , exemplifies the difference between the TREE-DIET and GRAPH-DIET problems, and motivates further work on the latter.

**Exact iterative strategy for the genomic search of ncRNAs**

In this final case study, we consider an exact filtering strategy to search new occurrences of a structured RNA within a given genomic context. In this setting, one attempts to find all  $\epsilon$ -admissible (cost  $\leq \epsilon$ ) occurrences/hits of a structured RNA  $S$  of length  $n$  within a given genome of length  $g \gg n$ , broken down in windows of length  $\kappa.n$ ,  $\kappa > 1$ . Classically, one would align  $S$  against individual windows, and report those

associated with an  $\epsilon$ -admissible alignment cost. This strategy would have an overall  $\Theta(g \cdot n^{tw+2})$  time

complexity, applying for instance the algorithm of [12].

Our instance simplification framework enables an alternative strategy, that incrementally filters out unsuitable windows based on models of increasing granularity. Indeed, for any given target sequence, the min alignment cost  $c_\delta$  obtained for a simplified instance of treewidth  $tw - \delta$  can be corrected (cf Additional file 1: Section D) into a lower bound  $c_\delta^*$  for the min alignment cost  $c_0^*$  of the full-treewidth instance  $tw$ . Any window such that  $c_\delta^* > \epsilon$  thus also obeys  $c_0^* > \epsilon$ , and can be safely discarded from the list of putative  $\epsilon$ -admissible windows, without having to perform a full-treewidth alignment. Given the exponential growth of the alignment runtime for increasing treewidth values (see Fig. 10, right) this strategy is expected to yield substantial runtime savings.

We used this strategy to search occurrences of the Twister ribozyme (PDBID 4OJI), a highly-structured ( $tw = 5$ ) 54nts RNA initially found in *O. sativa* (Asian rice) [54]. We targeted the *S. bicolor* genome (sorghum), focusing on a 10kb region centered on the 2,485,140 position of the 5th chromosome, where an instance of the ribozyme was suspected within an uncharacterized transcript (LOC110435504). The 4OJI sequence and structure were extracted from the 3D model as above, and included into a tree decomposition  $T_5$  (73 edges), simplified into  $T_4$  (71 edges),  $T_3$  (68 edges) and  $T_2$  (61 edges) using the tree-diet algorithm.

We aligned all tree decompositions against all windows of size 58nts using a 13nts offset, and measured the score and runtime of the iterative filtering strategy using a cost

cutoff  $\varepsilon = -5$ . The search recovers the suspected occurrence of twister as its best result (Fig. 11C), but produced hits (cf Fig. 11D) with comparable sequence conservation that could be the object of further studies. Regarding the filtering strategy, while  $\mathcal{T}_2$  only allows to rule out 3 windows out of 769,  $\mathcal{T}_3$  allows to eliminate an important proportion of putative targets, retaining only 109 windows, further reduced to 15 windows by  $\mathcal{T}_4$ , 6 of which end up as final hits for the full model  $\mathcal{T}_5$  (cf Fig. 11A). The search remains exact, but greatly reduces the overall runtime from 24 h to 34 min (42 fold!).

### Conclusion and discussion

We have established the parameterized complexity of three treewidth reduction problems, motivated by applications in Bioinformatics, as well as proposed practical algorithms for instances of reasonable treewidths. The reduced widths obtained by our proposed algorithm can be used to obtain: (i) sensitive heuristics, owing to the consideration of a maximal amount of edges/information in the thinned graphs; (ii) *a posteriori* approximation ratios, by comparing the potential contribution of removed edges to the optimal score obtained of the thinned instance by a downstream FPT/XP algorithm; (iii) substantial practical speedups without loss of correctness, e.g. when partial filtering can be safely achieved based on simplified input graphs.

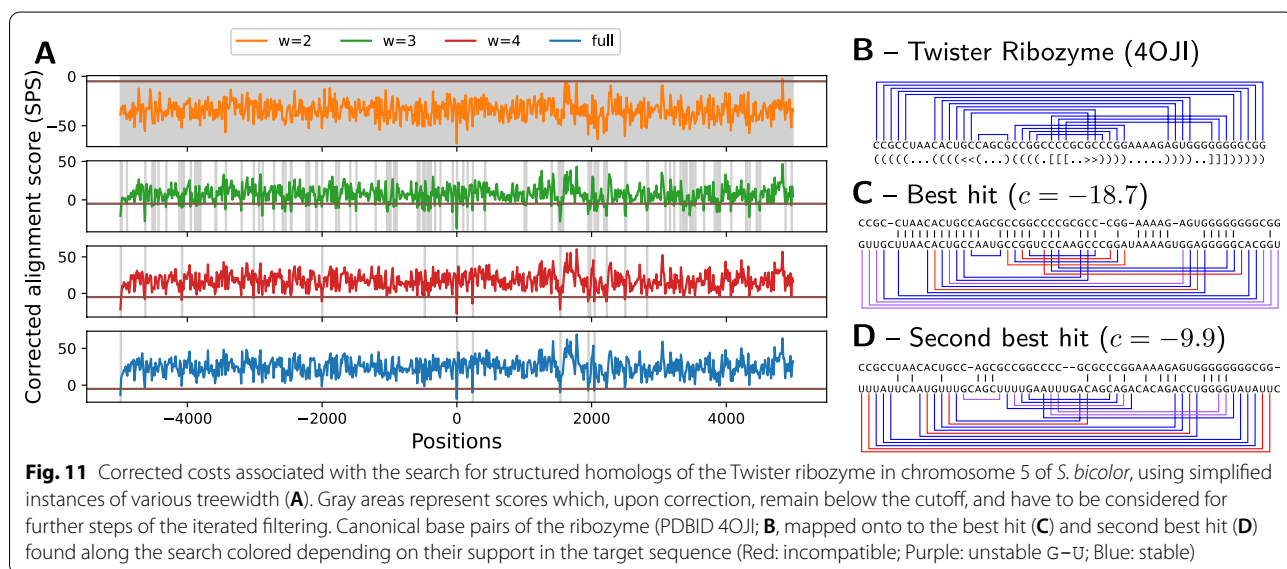
### Open questions

Regarding the parameterized complexity of GRAPH-DIET and TREE-DIET, some questions remain open (see Table 1): an FPT algorithm for TREE-DIET (ideally, with  $2^{O(tw)} \cdot n$  running time), would be the most desirable, if

possible satisfying the backbone constraints. The existence of such an algorithm is not trivial. In particular, it is perhaps worth noting that it is not implied by the existence of an FPT algorithm for GRAPH-DIET with the input treewidth as a parameter (1). Indeed, in comparison to the latter, TREE-DIET subtly restricts the search space to tree decompositions that are subsets of the input tree decomposition. It follows that the result of GRAPH DIET for a graph  $G$  may substantially differ from the result of TREE-DIET given a tree decomposition  $\mathcal{T}$  of  $G$  as input. We also aim at trying to give efficient exact algorithms for GRAPH DIET in the context of RNA (we conjecture this is impossible in the general case). Finally, we did not include the number of deleted edges in our multivariate analysis: even though in practice it is more difficult *a priori* to guarantee their small number, we expect it can be used to improve the running time in many cases.

### Backbone preservation

In two of our applications, the RNA secondary structure graph contains two types of edges: those representing the *backbone* of the sequence (i.e., between consecutive bases) and those representing base pair bonds. In practice, we want all backbone edges to be visible in the resulting tree-decomposition, and only base pairs may be lost. This can be integrated to the TREE-DIET model (and to our algorithms) using weighted edges, using the total weight rather than the count of deleted edges for the objective function. Note that some instances might be unrealizable (with no tree-diet preserving the backbone, especially for low  $tw'$ ). In most cases, ad-hoc bag duplications can help avoid this issue. The design of pre-processing methods, involving bag duplications or other



operations on tree decompositions, and aimed at ensuring the existence of a backbone-preserving tree-diet will be the subject of future work.

From a theoretical perspective, weighted edges may only increase the algorithmic complexity of the problems. However, a more precise model could consider graphs which already include a hamiltonian path (the backbone), and the remaining edges form a degree-one or two subgraph. Such extra properties may, in some cases, actually reduce the complexity of the problem. As an extreme case, we conjecture the PATH-DIET problem for  $tw' = 1$  becomes polynomial in this setting.

## Supplementary Information

The online version contains supplementary material available at <https://doi.org/10.1186/s13015-022-00213-z>.

**Additional file 1.** Supplementary sections: **A** Editing Trees before the Diet; **B** Pseudo-code; **C** Correctness of the rejection-based sampling of RNA designs; **D** Lower bound for the min. alignment cost from simplified models.

## Acknowledgements

The authors would like to thank Julien Baste for pointing out prior work on treewidth modulators, and providing valuable input regarding vertex deletion problems.

## Author's contributions

All authors contributed equally. All authors read and approved the final manuscript.

## Availability of data and materials

Source code of tree-diet method available at <https://gitlab.inria.fr/ambio/tree-diet>

## Declarations

### Competing interests

The authors declare that they have no competing interests.

Received: 15 November 2021 Accepted: 1 March 2022

Published online: 02 April 2022

## References

- Weller M, Chateau A, Giroudeau R. Exact approaches for scaffolding. *BMC Bioinformatics*. 2015; 16(S14). <https://doi.org/10.1186/1471-2105-16-s14-s2>
- Xu J. Rapid protein side-chain packing via tree decomposition. In: *Research in Computational Molecular Biology (RECOMB 2005)*. Lecture Notes in Computer Science. 2005; vol. 3500, pp. 423–439. Springer, Cambridge, USA. [https://doi.org/10.1007/11415770\\_32](https://doi.org/10.1007/11415770_32).
- Bulteau L, Fertin G, Jiang M, Rusu I. Tractability and approximability of maximal strip recovery. *Theor Comput Sci*. 2012;440:14–28.
- Baste J, Paul C, Sau I, Scornavacca C. Efficient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees. *Bull Math Biol*. 2017;79(4):920–38. <https://doi.org/10.1007/s11538-017-0260-y>.
- Bulteau L, Weller M. Parameterized algorithms in bioinformatics: an overview. *Algorithms*. 2019;12(12):256. <https://doi.org/10.3390/a12120256>.
- Waterman MS. Secondary structure of single stranded nucleic acids. *Adv Math Suppl Stud*. 1978;1(1):167–212.
- Xayaphoummine A, Bucher T, Thalmann F, Isambert H. Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations. *Proc Natl Acad Sci USA*. 2003;100(26):15310–5.
- Akutsu T. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Appl Math*. 2000;104(1–3):45–62. [https://doi.org/10.1016/S0166-218X\(00\)00186-4](https://doi.org/10.1016/S0166-218X(00)00186-4).
- Lyngsø RB, Pedersen CNS. RNA pseudoknot prediction in energy-based models. *J Comput Biol*. 2000;7(3–4):409–27.
- Sheikh S, Backofen R, Ponty Y. Impact Of The Energy Model On The Complexity Of RNA Folding With Pseudoknots. In: Kärkkäinen, J., Stoye, J. (eds.) *CPM - 23rd Annual Symposium on Combinatorial Pattern Matching*. *Combinatorial Pattern Matching*. 2012; vol. 7354, pp. 321–333. Springer, Helsinki, Finland. [https://doi.org/10.1007/978-3-642-31265-6\\_26](https://doi.org/10.1007/978-3-642-31265-6_26). Juha Kärkkäinen.
- Blin G, Denise A, Dulucq S, Herrbach C, Touzet H. Alignments of RNA structures. *IEEE/ACM Trans Comput Biol Bioinform*. 2010;7(2):309–22. <https://doi.org/10.1109/tcbb.2008.28>.
- Rinaudo P, Ponty Y, Barth D, Denise A. Tree decomposition and parameterized algorithms for rna structure-sequence alignment including tertiary interactions and pseudoknots. In: Raphael B, Tang J, editors. *Algorithms in Bioinformatics*. Ljubljana, Slovenia: Springer; 2012. p. 149–64.
- Kalvari I, Nawrocki EP, Ontiveros-Palacios N, Argasinska J, Lamkiewicz K, Marz M, Griffiths-Jones S, Toffano-Nioche C, Gautheret D, Weinberg Z, Rivas E, Eddy SR, Finn RD, Bateman A, Petrov AI. Rfam 14: expanded coverage of metagenomic, viral and microRNA families. *Nucleic Acids Res*. 2020;49(D1):192–200. <https://doi.org/10.1093/nar/gkaa1047>.
- Sarrazin-Gendron R, Yao H-T, Reinharz V, Oliver CG, Ponty Y, Waldspühl J. Stochastic sampling of structural contexts improves the scalability and accuracy of RNA 3d module identification. In: *Lecture Notes in Computer Science*. 2020; pp. 186–201. Springer, Padua, Italy. [https://doi.org/10.1007/978-3-030-45257-5\\_12](https://doi.org/10.1007/978-3-030-45257-5_12).
- Leontis NB, Westhof E. Geometric nomenclature and classification of RNA base pairs. *RNA*. 2001;7(4):499–512.
- Reinharz V, Soulé A, Westhof E, Waldspühl J, Denise A. Mining for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network families. *Nucleic Acids Res*. 2018;46(8):3841–51. <https://doi.org/10.1093/nar/gky197>.
- Gogate V, Dechter R. A complete anytime algorithm for treewidth. 2012; arXiv preprint [arXiv:1207.4109](https://arxiv.org/abs/1207.4109).
- Bodlaender HL, Koster AM. Treewidth computations i. upper bounds. *Informat Comput*. 2010;208(3):259–75.
- Song Y, Liu C, Malmberg R, Pan F, Cai L. Tree decomposition based fast search of RNA structures including pseudoknots in genomes. In: *Computational Systems Bioinformatics Conference, 2005. Proceedings*. 2005; 2005 IEEE, pp. 223–234. IEEE.
- Han B, Dost B, Bafna V, Zhang S. Structural alignment of pseudoknotted RNA. *J Comput Biol*. 2008;15(5):489–504. <https://doi.org/10.1089/cmb.2007.0214>.
- Vucinic J, Simoncini D, Ruffini M, Barbe S, Schiex T. Positive multistate protein design. *Bioinformatics*. 2019;36(1):122–30. <https://doi.org/10.1093/bioinformatics/btz497>.
- Yao H-T, Waldspühl J, Ponty Y, Will S. Taming disruptive base pairs to reconcile positive and negative structural design of RNA. In: *Research in Computational Molecular Biology. 25th International Conference on Research in Computational Molecular Biology (RECOMB 2021)*, Padua, France. 2021.
- Hammer S, Wang W, Will S, Ponty Y. Fixed-parameter tractable sampling for RNA design with multiple target structures. *BMC Bioinformatics*. 2019;20(1). <https://doi.org/10.1186/s12859-019-2784-7>.
- Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE. The protein data bank. *Nucleic Acids Res*. 2000;28:235–42. <https://doi.org/10.1093/nar/28.1.235>.
- Lu X-J, Bussemaker HJ, Olson WK. Dssr: an integrated software tool for dissecting the spatial structure of rna. *Nucleic Acids Res*. 2015;43(21):142–142.
- van Dijk T, van den Heuvel J-P, Slob W. Computing treewidth with libtw. Citeseer. <http://citeseerx.ist.psu.edu/viewdoc/download>. 2006.
- Bodlaender HL. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J Comput*. 1996;25(6):1305–17.
- Downey RG, Fellows MR. *Parameterized complexity*. Berlin: Springer; 2012.



29. Cygan M, Fomin FV, Kowalik Ł, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S. Parameterized algorithms, vol. 5. Cham: Springer; 2015.
30. El-Mallah ES, Colbourn CJ. The complexity of some edge deletion problems. *IEEE Trans Circ Syst*. 1988;35(3):354–62.
31. Crespelle C, Drange PG, Fomin FV, Golovach PA. A survey of parameterized algorithms and the complexity of edge modification. 2020; arXiv preprint [arXiv:2001.06867](https://arxiv.org/abs/2001.06867).
32. Cai L. Parameterized complexity of vertex colouring. *Discrete Appl Math*. 2003;127(3):415–29.
33. Lovász L. Graph minor theory. *Bull Am Math Soc*. 2006;43(1):75–86.
34. Robertson N, Seymour PD. Graph minors. xiii. the disjoint paths problem. *J Combinat Theo Ser B*. 1995;63(1):65–110.
35. Cygan M, Lokshtanov D, Pilipczuk M, Pilipczuk M, Saurabh S. On the hardness of losing width. In: *International Symposium on Parameterized and Exact Computation*. 2011; pp. 159–168. Springer
36. Baste J, Sau I, Thilikos DM. Hitting minors on bounded treewidth graphs. i. general upper bounds. *SIAM J Discret Math*. 2020;34(3):1623–48. <https://doi.org/10.1137/19M1287146>.
37. Courcelle B. The monadic second-order logic of graphs iii: Tree-decompositions, minors and complexity issues. *RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications*. 1992;26(3):257–86.
38. Arnborg S, Lagergren J, Seese D. Easy problems for tree-decomposable graphs. *J Algo*. 1991;12(2):308–40.
39. Saitoh T, Yoshinaka R, Bodlaender HL. Fixed-treewidth-efficient algorithms for edge-deletion to interval graph classes. In: *Algorithms and Computation-15th International Conference and Workshops (WALCOM 2021)*. Lecture Notes in Computer Science. 2021; vol. 12635, pp. 142–153. Springer, Yangon, Myanmar. [https://doi.org/10.1007/978-3-030-68211-8\\_12](https://doi.org/10.1007/978-3-030-68211-8_12).
40. Tan J, Zhang L. The consecutive ones submatrix problem for sparse matrices. *Algorithmica*. 2007;48(3):287–99.
41. Proskurowski A, Telle JA. Classes of graphs with restricted interval models. *Discret Math Theor Comput Sci*. 2006; **3**(4)
42. Bodlaender HL, Koster AM. Combinatorial optimization on graphs of bounded treewidth. *Comput J*. 2008;51(3):255–69.
43. Bodlaender HL. Discovering treewidth. In: *International Conference on Current Trends in Theory and Practice of Computer Science*. 2005; pp. 1–16. Springer
44. Jakob W, Rhineland J, Moldovan D. pybind11—Seamless operability between C++11 and Python. <https://github.com/pybind/pybind11>. 2017.
45. Klein RJ, Eddy SR. Rsearch: finding homologs of single structured RNA sequences. *BMC Bioinform*. 2003;4(1):44.
46. Rivas E, Eddy SR. Parameterizing sequence alignment with an explicit evolutionary model. *BMC Bioinform*. 2015;16(1):406.
47. Wang W. Practical sequence-structure alignment of rnas with pseudoknots. PhD thesis, Université Paris-Saclay, School of Computer Science. 2017.
48. Wang W, Denise A, Ponty Y. LicoRNA: aLignment of Complex RNAs v1.0. 2017; <https://licorna.lri.fr>.
49. Sudarsan N, Lee ER, Weinberg Z, Moy RH, Kim JN, Link KH, Breaker RR. Riboswitches in eubacteria sense the second messenger cyclic di-gmp. *Science*. 2008;321(5887):411–3. <https://doi.org/10.1126/science.1159519>. <https://science.sciencemag.org/content/321/5887/411.full.pdf>.
50. Tamayo R. Cyclic diguanylate riboswitches control bacterial pathogenesis mechanisms. *PLOS Pathogens*. 2019;15(2):1–7. <https://doi.org/10.1371/journal.ppat.1007529>.
51. Smith KD, Shanahan CA, Moore EL, Simon AC, Strobel SA. Structural basis of differential ligand recognition by two classes of bis-(3'-5')-cyclic dimeric guanosine monophosphate-binding riboswitches. *Proc Nat Acad Sci*. 2011;108(19):7757–62. <https://doi.org/10.1073/pnas.1018857108>. <https://www.pnas.org/content/108/19/7757.full.pdf>
52. Lu X-J, Bussemaker HJ, Olson WK. DSSR: an integrated software tool for dissecting the spatial structure of RNA. *Nucleic Acids Res*. 2015;43(21):142–142. <https://doi.org/10.1093/nar/gkv716>. <https://academic.oup.com/nar/article-pdf/43/21/e142/17435026/gkv716.pdf>.
53. Thompson JD, Plewniak F, Poch O. BALiBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*. 1999;15(1):87–8.
54. Liu Y, Wilson TJ, McPhee SA, Lilley DM. Crystal structure and mechanistic investigation of the twister ribozyme. *Nat Chem Biol*. 2014;10(9):739–44.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more [biomedcentral.com/submissions](https://biomedcentral.com/submissions)

