



**HAL**  
open science

# Tree Diet: Reducing the Treewidth to Unlock FPT Algorithms in RNA Bioinformatics

Bertrand Marchand, Yann Ponty, Laurent Bulteau

► **To cite this version:**

Bertrand Marchand, Yann Ponty, Laurent Bulteau. Tree Diet: Reducing the Treewidth to Unlock FPT Algorithms in RNA Bioinformatics. WABI 2021 - 21st Workshop on Algorithms in Bioinformatics, 2021, Paris, France. hal-03206132

**HAL Id: hal-03206132**

**<https://inria.hal.science/hal-03206132>**

Submitted on 23 Apr 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Tree Diet: Reducing the Treewidth to Unlock FPT Algorithms in RNA Bioinformatics

Bertrand Marchand 

LIX CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

LIGM, CNRS, Univ Gustave Eiffel, F77454 Marne-la-vallée, France

bertrand.marchand@lix.polytechnique.fr

Yann Ponty<sup>1</sup> 

LIX CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

yann.ponty@lix.polytechnique.fr

Laurent Bulteau<sup>1</sup> 

LIGM, CNRS, Univ Gustave Eiffel, F77454 Marne-la-vallée, France

laurent.bulteau@u-pem.fr

---

## 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.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Dynamic programming; Theory of computation  $\rightarrow$  Parameterized complexity and exact algorithms; Applied computing  $\rightarrow$  Bioinformatics

**Keywords and phrases** RNA, treewidth, FPT algorithms, RNA design, structure-sequence alignment

**Digital Object Identifier** 10.4230/LIPIcs.WABI.2021.

**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.

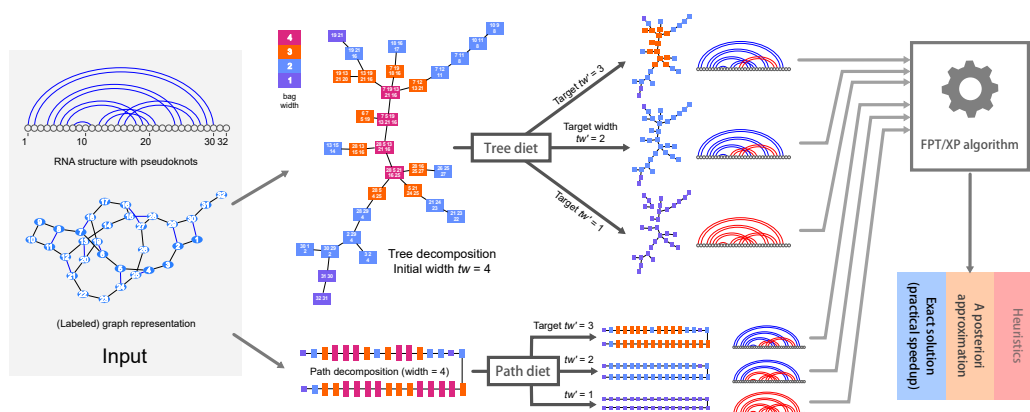
## 1 Introduction

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 [40], structural bioinformatics [42], comparative genomics [7], phylogenetics [2], and further examples that can be found in a review by Bulteau and Weller [8]. RNA bioinformatics is no exception, with the prevalence of the secondary structure, an outer planar graph [39], as an abstraction of RNA conformations, and the notable utilization of

---

<sup>1</sup> To whom correspondence should be addressed





■ **Figure 1** General description of our approach and rationale. Starting from a structured instance, *e.g.* an RNA structure with pseudoknots, our tree diet/path diet algorithms extract simplified tree/path decompositions, having prescribed target width  $tw'$ . Those can be used within existing parameterized algorithms to yield efficient heuristics, *a posteriori* approximations or even exact solutions.

43 graph models to represent complex topological motifs called pseudoknots [41], inducing the  
 44 hardness of several tasks, such as structure prediction [1, 23, 29], structure alignment [5], or  
 45 structure/sequence alignment [25]. Such motifs are functionally important and conserved, as  
 46 witnessed by their presence in the consensus structure of 336 RNA families in the 14.5 edition  
 47 of the Rfam database [18]. Moreover, methods in RNA bioinformatics [28] are increasingly  
 48 considering non-canonical base pairs and modules [20, 24], further increasing the density of  
 49 RNA structural graphs and outlining the need for scalable algorithms.

50 A parameterized complexity approach can be used to circumvent the frequent NP-  
 51 hardness of relevant problems. It generally considers one or several parameters, whose  
 52 values are naturally bounded (or much smaller than the length) within real-life instances.  
 53 Once identified, one aims to design a Fixed Parameter Tractable (FPT) algorithm, having  
 54 polynomial complexity for any fixed value of the parameter, and reasonable dependency on  
 55 the parameter value. The treewidth is a classic parameter for FPT algorithms, and intuitively  
 56 captures a notion of distance of the input to a tree. It is popular in bioinformatics due to the  
 57 existence of exact FPT algorithms (and efficient heuristics) for computing a tree-decomposition  
 58 that minimizes the treewidth. Given a tree-decomposition, many combinatorial optimization  
 59 tasks can be solved using dynamic programming (DP), in time/space complexities that  
 60 remain polynomial for any fixed treewidth value. Resulting algorithms remain correct upon  
 61 (almost) arbitrary modifications of the objective function parameters, and can be adapted to  
 62 study statistical properties of search spaces through changes of algebra.

63 Unfortunately, the existence of a parameterized (or FPT) algorithm does not necessarily  
 64 imply that of a practically-efficient implementation, even when the parameter takes low  
 65 typical values. Indeed, the dependency of the complexity on the treewidth may be prohibitive,  
 66 both in terms of time and memory requirements. This limitation is particularly obvious while  
 67 searching and aligning structured RNAs, giving rise to an algorithmic problem called the  
 68 RNA structure-sequence alignment [31, 16, 25], for which the best known exact algorithm  
 69 is in  $\Theta(n \cdot m^{tw+1})$ , where  $n$  is the structure length,  $m$  the sequence/window, and  $tw$  is the  
 70 treewidth of the structure (inc. backbone). Similar complexities hold for problems that  
 71 can be expressed as (weighted) constraint satisfaction problems, with  $m$  representing the  
 72 cardinality of the variable domains. Such frameworks are frequently used for molecular

73 design, both in proteins [36] and RNA [43], and may require the consideration of tree-widths  
74 up to 20 or more [15].

75 In this paper, we investigate a pragmatic strategy to increase the practicality of para-  
76 meterized algorithms based on the treewidth parameter [6]. We put our instance graphs  
77 on a diet, *i.e.* we introduce a preprocessing that reduces their treewidth to a prescribed  
78 value by removing a minimal cardinality/weight set of edges. As shown above, the prac-  
79 tical complexity of many algorithms greatly benefits from the consideration of simplified  
80 instances, having lower treewidth. Moreover, specific countermeasures can sometimes be  
81 used to preserve the correctness of the algorithm. For instance, searching structured RNAs  
82 using RNA structure-sequence alignment [31], a iterated filtering strategy could use instances  
83 of increasing treewidth to restrict potential hits, weeding them early so that a – costly –  
84 full structure is reserved to (quasi-)hits. This strategy could remain exact while saving  
85 substantial time. Alternative countermeasures could be envisioned for other problems, such  
86 as a rejection approach to correct a bias introduced by simplified instances in RNA design.

87 After stating our problem(s) in Section 2, we study in Section 3 the parameterized  
88 complexity of the GRAPH-DIET problem, the removal of edges to reach a prescribed treewidth.  
89 We propose in Section 4 two DP practical FPT algorithms for TREE-DIET and PATH-  
90 DIET, two natural simplifications of the GRAPH-DIET problem, where a tree (resp. path)  
91 decomposition is provided as input and used as a guide. Finally, we show in Section 5 how  
92 our algorithm can be used to extract hierarchies of graphs/structural models of increasing  
93 complexity to provide alternative sampling strategies for RNA design, and speed-up the search  
94 for pseudoknotted non-coding RNAs. We conclude in Section 6 with future considerations  
95 and open problems.

## 96 2 Statement of the problem(s) and results

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

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

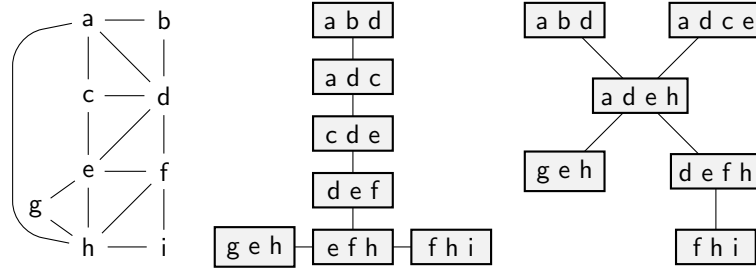
106 A *tree diet* of  $\mathcal{T}$  is any tree-decomposition  $\mathcal{T}'$  obtained by removing vertices from the  
107 bags of  $\mathcal{T}$ .  $\mathcal{T}'$  is a  *$d$ -tree diet* if  $w(\mathcal{T}') \leq w(\mathcal{T}) - d$ .

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

110 We define BINARY-TREE-DIET and PATH-DIET analogously, where  $\mathcal{T}$  is restricted to be  
111 a binary tree (respectively, a path). An example is given in Figure 2.

## 112 Parameterized Complexity in a Nutshell

113 The basics of parameterized complexity can be loosely defined as follows (see [13] for the  
114 formal background). A *parameter  $k$*  for a problem is an integer associated with each instance  
115 which is expected to remain small in practical instances (especially when compared to the



■ **Figure 2** Illustrations for the GRAPH-DIET and TREE-DIET problems. Given a graph  $G$  on the left (treewidth 3), an optimal solution for GRAPH-DIET yields the tree-decomposition in the middle (width 2, edge  $ah$  is lost). On the other hand, any 1-tree diet for the tree-decomposition on the right loses at least 3 edges

116 input size  $n$ ). An exact algorithm, or the problem it solves, is FPT if it takes time  $f(k)\text{poly}(n)$ ,  
 117 and XP if it takes time  $n^{g(k)}$  (for some functions  $f, g$ ). Under commonly accepted conjectures,  
 118 W[1]-hard problems may not be FPT, and Para-NP-hard problems (NP-hard even for some  
 119 fixed value of  $k$ ) may not be FPT nor XP.

## 120 2.1 Our results

121 Our results are summarized in Table 1. Although the GRAPH-DIET problem would give the  
 122 most interesting tree-decompositions in theory, it seems unlikely to admit efficient algorithms  
 123 in practice (see Section 3.1).

124 Thus we focus on the TREE-DIET relaxation, where an input tree-decomposition is given,  
 125 which we use as a guide / a restriction towards a thinner tree-decomposition. Seen as an  
 126 additional constraint, it makes the problem harder (the case  $tw' = 1$  becomes NP-hard,  
 127 Theorem 2, although for GRAPH-DIET it corresponds to Spanning Tree and is polynomial),  
 128 however it does help reduce the search space, especially with parameter  $tw$ . In Theorem 8 we  
 129 give an  $O((6\Delta)^{tw}\Delta^2n)$  Dynamic Programming algorithm, where  $\Delta$  is the maximum number  
 130 of children of any bag in the tree-decomposition. This algorithm can thus be seen as XP in  
 131 general, but FPT on bounded-degree tree-decompositions (e.g. binary trees and paths). This  
 132 is not a strong restriction, since the input tree may safely and efficiently be transformed into  
 133 a binary one (see Appendix A for more details).

134 We also consider the case where the treewidth needs to be reduced by  $d = 1$  only,  
 135 this without constraining the source treewidth. We give a polynomial-time algorithm for  
 136 PATH-DIET in this setting (Theorem 11) which generalizes into an XP algorithm for larger  
 137 values of  $d$ , noting that an FPT algorithm for  $d$  is out of reach by Theorem 4. We also show  
 138 that the problem is NP-hard if the tree degree is unbounded (Theorem 3).

## 139 3 Algorithmic Limits: Parameterized Complexity Considerations

### 140 3.1 The intricate Graph-Diet Problem

141 GRAPH-DIET can be seen as a special case of the EDGE DELETION PROBLEM (EDP) for the  
 142 family of graphs  $\mathcal{H}$  of treewidth  $tw'$ : given a graph  $G$ , remove as few edges as possible to  
 143 obtain a graph in  $\mathcal{H}$ . Such edge modification problems are more often parameterized by the  
 144 number  $k$  of edited edges (see [11] for a complete survey). Given our focus on increasing  
 145 the practicality of treewidth-based algorithms in bioinformatics, we restrict our focus to  
 146 treewidth related parameters  $tw, tw'$  and  $d = tw - tw'$ .

Parameter Problem	Source treewidth $tw$		Target treewidth $tw'$	Difference $d = tw - tw'$		
GRAPH-DIET	<i>open</i>		Para-NP-hard $tw' = 2$ EDP( $K_4$ ) [14]	Para-NP-hard* $d = 1$ Theorem 1		
TREE-DIET	XP $O^*((6\Delta)^{tw})$ Theorem 8	FPT <i>open</i>	Para-NP-hard $tw' = 1$ Theorem 2	Para-NP-hard $d = 1$ Theorem 3		
BINARY-TREE-DIET	FPT $O^*(12^{tw})$ Theorem 8			W[1]-hard Theorem 4	XP <i>open</i>	
PATH-DIET					XP $O^*(tw^d)$ Theorem 11	

■ **Table 1** Parameterized results for our problems. Algorithm complexities are given up to polynomial time factors,  $\Delta$  denotes the maximum number of children in the input tree-decomposition. (\*) see Theorem 1 statement for a more precise formulation.

147 Considering the target treewidth  $tw'$ , we note that EDP is NP-hard when  $\mathcal{H}$  is the family  
 148 of treewidth-2 graphs [14], namely  $K_4$ -free graphs, hence the notation EDP( $K_4$ ). It follows  
 149 that GRAPH-DIET is Para-NP-hard for the target treewidth parameter  $tw'$ .

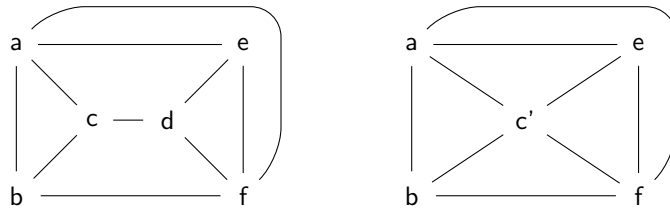
150 Regarding the source graph treewidth  $tw$ , and as it is often the case with treewidth-based  
 151 problems, graph minor theorems may "freely" give parameterized algorithms. In this respect,  
 152 GRAPH-DIET corresponds to deciding if a graph  $G$  belongs to the family  $Treewidth-tw'+ke$ ,  
 153 representing the graphs of treewidth  $tw'$ , augmented by  $k$  additional edges. However, this  
 154 family is not minor-closed: for some graphs in the family, there is an edge contraction that  
 155 yields a new graph  $G'$  that is not in  $Treewidth-tw'+ke$ , as illustrated by Figure 3.

156 Another theoretical approach is via Courcelle's Theorem [10] and Monadic Second Order  
 157 (MSO) formulas: it is sufficient to express the property of being in  $Treewidth-tw'+ke$  using  
 158 MSO to prove that GRAPH-DIET is FPT for  $tw$ . We presume this is feasible (the main brick  
 159 being minor testing, which is expressible in MSO), however it is not clear whether this is  
 160 doable with formulas independent of  $k$ , in order to obtain an algorithm for the treewidth  
 161 alone. In any case, this approach would probably not yield practical algorithms. Indeed,  
 162 Courcelle's theorem typically lead to running times involving towers of exponentials on the  
 163 relevant parameters, so we do not investigate it further within the scope of this paper.

164 Another meta-theorem by Cai [9] may yield an FPT algorithm if  $Treewidth-tw'+ke$  can  
 165 be described through forbidden induced subgraphs, but again  $k$  would most likely be a  
 166 parameter as well. On a related note, it is worth noting that Edge Deletion to other graph  
 167 classes (interval, permutation, ...) does admit efficient algorithms when parameterized by  
 168 the treewidth alone [27], painting a contrasted picture.

169 The vertex deletion equivalent of GRAPH-DIET, where one asks for a minimum subset  
 170 of vertices to remove to obtain a given treewidth, is known as a TREewidth MODULATOR.  
 171 This problem has been better-studied than its edge-deletion counterpart [12], and has been  
 172 shown to be FPT for the treewidth [3], with a reasonable dependency in the parameter.  
 173 However, it is currently unclear how this can be adapted into an edge deletion algorithm.

174 Overall an FPT algorithm for GRAPH-DIET does not seem out of reach, and could result  
 175 from one of the above-mentioned meta-theorems. However it seems unlikely to induce a  
 176 "practical" exact algorithms. Indeed, any algorithm for GRAPH-DIET can be used to compute  
 177 the TREewidth of an arbitrary graph, for which current state-of-the-art exact algorithms



■ **Figure 3** A graph  $G$  (left) with treewidth 3. Deleting edge  $cd$  gives treewidth 2. However, if one contracts edge  $cd$ , then the resulting graph (right) has treewidth 3, and deleting any single edge does not decrease the treewidth. This example shows that the graph family **Treewidth 2+1e** is not minor-closed.

178 require time in  $tw^{O(tw^3)}$  [6]. We thus have the following conjecture, which motivates the  
 179 **TREE-DIET** relaxation of the problem.

180 ► **Conjecture 1.** **GRAPH-DIET** is FPT for the source treewidth parameter ( $tw$ ), but no  
 181 algorithm with single-exponential running time exists.

182 Finally, for parameter  $d$ , any polynomial-time algorithm for constant  $d$  would allow to  
 183 compute the treewidth of any graph in polynomial time, so we have the following result (see  
 184 Appendix B for more details).

185 ► **Theorem 1.** There is no XP algorithm for **GRAPH-DIET** with parameter  $d$  unless  $P = NP$ .

### 186 3.2 Lower Bounds for Tree-Diet

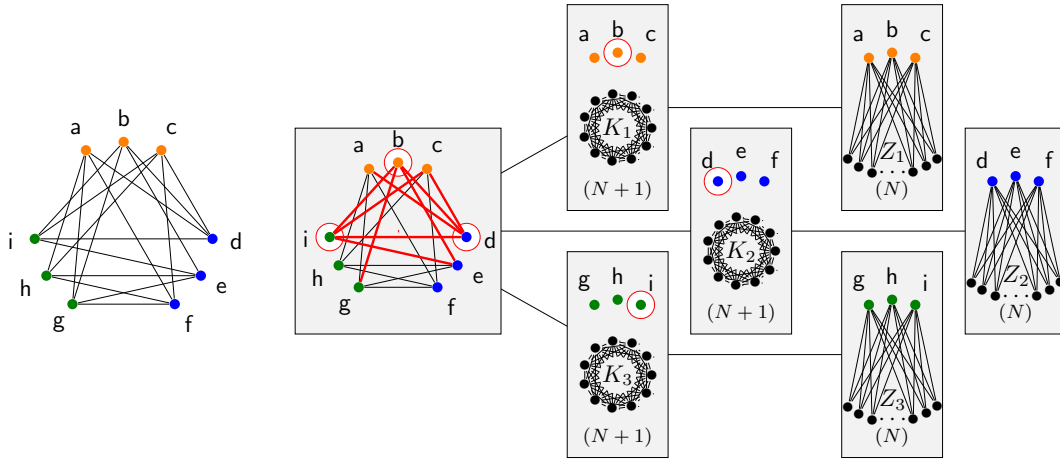
187 Parameters  $tw'$  and  $d$  would be the most interesting in practice, since parameterized algorithms  
 188 would be efficient for small diets or small target treewidth. However, we prove strong lower-  
 189 bounds for **TREE-DIET** on each of these parameters, leaving very little hope for parameterized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

193 ► **Theorem 2.** **TREE-DIET** and **PATH-DIET** are Para-NP-hard for the target treewidth  
 194 parameter  $tw'$  (i.e. NP-hard for  $tw' = 1$ ).

195 **Proof.** By reduction from the NP-hard problem **SPANNING CATERPILLAR TREE** [34]: given  
 196 a graph  $G$ , does  $G$  has a spanning tree  $C$  that is a caterpillar? Given  $G = (V, E)$  with  
 197  $n = |V|$ , we build a tree-decomposition  $\mathcal{T}$  of  $G$  consisting of  $n - 1$  bags containing all vertices  
 198 connected in a path. Then  $(G, \mathcal{T})$  admit a tree diet to treewidth 1 with  $n - 1$  visible edges if,  
 199 and only if,  $G$  admits a caterpillar spanning tree. Indeed, the subgraph of  $G$  with visible  
 200 edges must be a graph with pathwidth 1, i.e. a caterpillar. With  $n - 1$  visible edges, the  
 201 caterpillar connects all  $n$  vertices together, i.e. it is a spanning tree. ◀

202 ► **Theorem 3.** **TREE-DIET** is Para-NP-hard for parameter  $d$ . More precisely, it is **W[1]-hard**  
 203 for parameter  $\Delta$  even when  $d = 1$ .

**Proof.** By reduction from **MULTI-COLORED CLIQUE**. 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 **W[1]-hard** under these restrictions [13]). Let  
 $L := \delta k - \binom{k}{2}$  and  $N = \max\{|V|, L + 1\}$ . We now build a graph  $G'$  and a tree-decomposition



■ **Figure 4** Reduction for Theorem 3 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  $G$ .

$\mathcal{T}'$ : 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  $T_0$  for each  $i \in [k]$ . Thus,  $\mathcal{T}$  is a tree-decomposition of  $G'$  with  $\Delta = k - 1$  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.

204  $\Leftarrow$  Assume  $G$  has a  $k$ -clique  $X = \{x_1, \dots, x_k\}$  (with  $x_i \in V_i$ ). Build  $\mathcal{T}'$  by removing  
 205 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  
 206 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  
 207 edges of  $G$  adjacent to any  $x_i$ . Since  $X$  form a clique and each  $x_i$  has degree  $d$ , there are  
 208  $L = kd - \binom{k}{2}$  such edges.

209  $\Rightarrow$  Consider a 1-tree diet  $\mathcal{T}'$  of  $\mathcal{T}$  losing  $L$  edges. Since each bag  $T_{1,i}$  has maximum  
 210 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  
 211  $x_i \in K_i$  would loose at least  $N \geq L + 1$  edges). Furthermore,  $x_i$  may not be removed from  
 212  $T_{2,i}$  (otherwise  $N$  edges between  $x_i$  and  $Z_i$  would be lost), so  $x_i$  must also be removed from  
 213  $T_0$ . Let  $K$  be the number of edges between two distinct  $x_i$ . The total number of lost edges  
 214 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  
 215  $G$ . ◀

216 ▶ **Theorem 4.** PATH-DIET is  $W[1]$ -hard for parameter  $d$ .

217 By a straightforward reduction from Clique, see Appendix B for more details.



218 **4 FPT Algorithm**219 **4.1 For general tree-decompositions**

220 We describe here a  $O(3^{tw}n)$ -space,  $O(\Delta^{tw+2} \cdot 6^{tw}n)$ -time dynamic programming algorithm for  
 221 the TREE-DIET problem, with  $\Delta$  and  $tw$  being respectively the maximum number of children  
 222 of a bag in the input tree-decomposition and its width. On *binary* tree-decompositions (where  
 223 each bag has at most 2 children), it yields a  $O(3^{tw}n)$ -space  $O(12^{tw}n)$ -time FPT algorithm.

224 **4.1.1 Coloring formulation**

225 We aim at solving the following problem: given a tree-decomposition  $\mathcal{T}$  of width  $tw$  of a  
 226 graph  $G$ , we want to remove vertices from the bags of  $\mathcal{T}$  in order to reach a target width  $tw'$ ,  
 227 while *losing* as few edges from  $G$  as possible. We tackle the problem through an equivalent  
 228 *coloring* formulation: our algorithm will assign a color to each occurrence of a vertex in a  
 229 bag. We work with three colors: red, orange and green. Green means that the vertex is kept  
 230 in the bag, while orange and red means removal. To ensure equivalence with the original  
 231 problem (specifically, that bags where a given vertex is green form a connected subtree), the  
 232 colors will be assigned following local rules, which we now describe.

233 We first root the tree-decomposition arbitrarily. A coloring of vertices in the bags of the  
 234 decomposition is said to be *valid* if it follows, when going down the tree, the following rules:

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

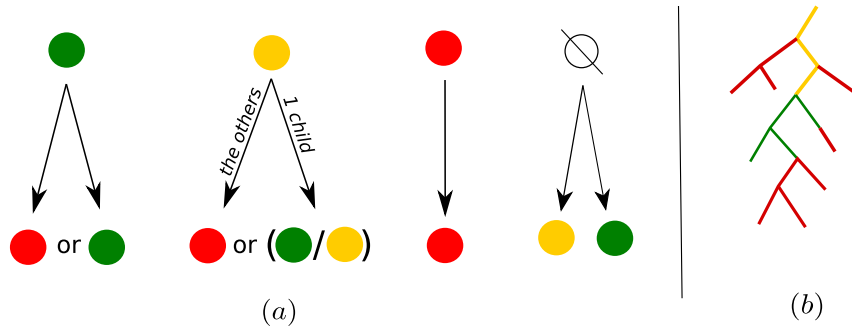
240 These rules are summarized in Figure 5 (a).

241 Since the output tree  $\mathcal{T}'$  must form a valid tree-decomposition, for each vertex  $u$ , the set  
 242 of bags containing  $u$  must form a (possibly empty) connected sub-tree within  $\mathcal{T}'$ . The rules  
 243 verifiably enforce that green occurrences of a vertex form a connected sub-tree. Informally,  
 244 orange vertices are locally absent but “may potentially be found further down the tree”,  
 245 while red vertices are removed from both the current bag and its entire sub-tree. Figure 5 (b)  
 246 shows an example sketch for a valid coloring of the occurrences of a given vertex in the  
 247 tree-decomposition. A vertex may only be orange along a path starting from its highest  
 248 occurrence in the tree, with any part branching off that path entirely red. It ends at the top  
 249 of a (potentially empty) green sub-tree, whose vertices may also be parents to entirely red  
 250 sub-trees.

251 We will now more formally prove the equivalence of the coloring formulation to the  
 252 original problem. Let us first introduce two definitions. Given a valid coloring  $\mathcal{C}$  of a  
 253 tree-decomposition of  $G$ , an edge  $(u, v)$  of  $G$  is said to be *realizable* if there exists a bag in  
 254 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  
 255  $d$ -*diet-valid* if removing red/orange vertices reduces the width of  $\mathcal{T}$  from  $w(\mathcal{T})$  to  $w(\mathcal{T}) - d$ .

256 ► **Proposition 5.** *Given a graph  $G$ , a tree-decomposition  $\mathcal{T}$  of width  $tw$ , and a target width*  
 257  *$tw' < tw$ , The TREE-DIET problem is equivalent to finding a  $(tw - tw')$ -diet-valid coloring  $\mathcal{C}$*   
 258 *of  $\mathcal{T}$  allowing for a number of realizable edges in  $G$  as large as possible.*

259 **Proof.** Given a  $(tw - tw')$ -tree-diet of  $\mathcal{T}$ , specifying which vertices are removed from which  
 260 bags, we obtain a valid coloring  $\mathcal{C}$  for  $\mathcal{T}$  incurring the same number of lost (unrealizable)  
 261 edges. To start with, a vertex  $u$  is colored green in the bags where it is not removed. By the



■ **Figure 5** (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 sub-tree of bags containing a given vertex  $u$

262 validity of  $\mathcal{T}'$  as a decomposition, this set of bags forms a connected sub-tree, that we denote  
 263  $\mathcal{T}_u^g$ . We also write  $\mathcal{T}_u$  for the sub-tree of bags containing  $u$  in the original decomposition  $\mathcal{T}$ .  
 264 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  
 265 the root of  $\mathcal{T}_u$  (included) and the root of  $\mathcal{T}_u^g$  (excluded).  $u$  is colored red in any other bag of  
 266  $\mathcal{T}_u$  not covered by these two cases. The resulting coloring follows rules (R1-4) and induces  
 267 the same set of lost/non-realizable edges as the original  $(tw - tw')$ -tree-diet. Conversely, an  
 268 equivalent  $(tw - tw')$ -tree-diet is obtained from a  $(tw - tw')$ -diet-valid coloring by removing  
 269 red/orange vertices and keeping green ones. ◀

#### 270 4.1.2 Decomposition of the search space and sub-problems

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

275 Consider an edge  $(u, v)$  of  $G$ , realizable when coloring a tree-decomposition  $\mathcal{T}$  of  $G$  with  
 276  $\mathcal{C}$ , 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  
 277 sub-tree-decomposition rooted at  $X_i$ ,  $G_i$  the subgraph induced by the vertices present in  
 278  $\mathcal{T}_i$ , and  $C(i, f)$  the colorings of  $\mathcal{T}_i$  agreeing with  $f$  on  $i$ . Our dynamic programming table is  
 279 then defined as:

$$280 \quad c(X_i, f) = \max_{\mathcal{C} \in C(i, f)} \left| \left\{ \begin{array}{l} \text{Edges } (u, v) \text{ of } G_i, \text{ realizable within } \mathcal{T}_i \text{ colored with } \mathcal{C} \\ \text{such that } \mathcal{T}_{uv}^g \text{ is entirely contained strictly below } X_i \end{array} \right\} \right|$$

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 coloring  $\mathcal{C}$  of  $\mathcal{T}_i$ , such that  $\mathcal{T}_{uv}^g$  is entirely contained strictly below  $X_i$ . We have:

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

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

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

288 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:

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

289 with

290 ■  $m$ : a map from the orange vertices in  $X_i$  to the children of  $X_i$ . It decides for each orange  
291 vertex  $u$ , which child, among those which contain  $u$ , will color  $u$  orange or green;

292 ■  $\text{comp}(Y_j, f, m)$ : the set of colorings of  $Y_j$  compatible with  $f$  on  $X_i$  and  $m$ ;

293 ■  $\text{count}(f, f'_j)$ : set of edges of  $G$  involving two vertices of  $Y_j$  green by  $f'_j$ , but such that one  
294 of them is either not in  $X_i$  or not green by  $f$ .

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

298 ► **Lemma 7.** An edge  $(u, v)$  of  $G$ , realizable in  $\mathcal{T}$  under  $\mathcal{C}$ , is contained in exactly one set of  
299 the form  $\text{count}(C_{|P}, C_{|X})$  with  $X$  a bag of  $\mathcal{T}$  and  $P$  its parent,  $C_{|P}, C_{|X}$  the restrictions of  $\mathcal{C}$   
300 to  $P$  and  $X$ , respectively, and “count” defined as above. In addition,  $X$  is the root of the  
301 sub-tree of  $\mathcal{T}$  in which both  $u$  and  $v$  are green.

302 The proofs of Theorem 6 and Lemma 7 are postponed to Appendix C.

### 303 4.1.2.1 Dynamic programming algorithm

304 The recurrence relation of Theorem 6 naturally yields a dynamic programming algorithm for  
305 the TREE-DIET problem, as stated below:

306 ► **Theorem 8.** There exists a  $O(\Delta^{tw+2} \cdot 6^{tw} \cdot n)$ -time,  $O(3^{tw} \cdot n)$ -space algorithm for the  
307 TREE-DIET problem, with  $\Delta$  the maximum number of children of a bag in the input tree-  
308 decomposition, and  $tw$  its width.

309 The proof of this theorem is postponed to Appendix C.

310 ► **Corollary 9.** BINARY-TREE-DIET ( $\Delta = 2$ ) admits an FPT algorithm for the  $tw$  parameter.

311 A pseudo-code implementation of the algorithm, using memoization, is included in Appendix E

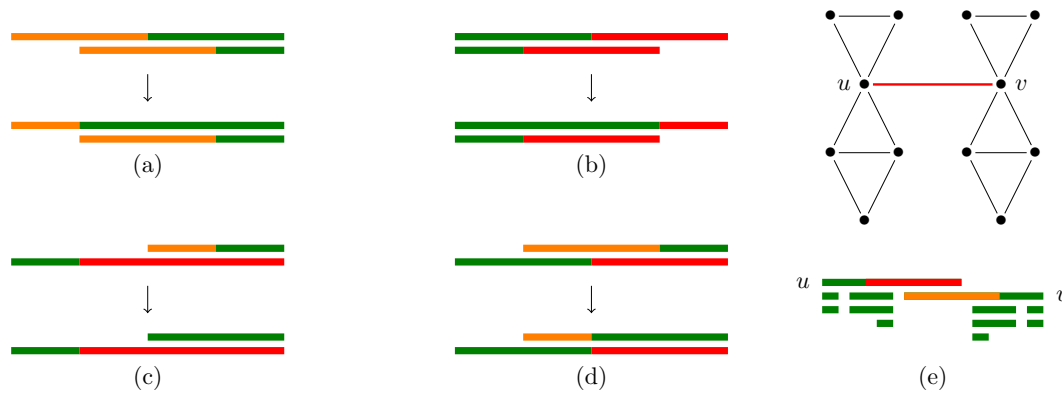
## 312 4.2 For path decompositions

313 In the context of paths decompositions, we note that the number of removed vertices per  
314 bag can be restricted limited to at most  $2d$  without losing the optimality. More precisely,  
315 we say that a coloring  $\mathcal{C}$  is  $d$ -simple if any bag has at most  $d$  orange and  $d$  red vertices. We  
316 obtain the following result, using transformations given in Figure 6.

317 ► **Proposition 10.** Given a graph  $G$  and a path-decomposition  $\mathcal{T}$ , if  $\mathcal{C}$  is a  $d$ -diet-valid  
318 coloring of  $\mathcal{T}$  losing  $k$  edges, then  $\mathcal{T}$  has a  $d$ -diet-valid coloring that is  $d$ -simple, and loses at  
319 most  $k$  edges.

320 The proof of this result is postponed to Appendix D. Together with Proposition 5, this shows  
321 that it is sufficient to restrict our algorithm to  $d$ -simple colorings. (See also Figure 6). In  
322 particular, for any set  $X_i$ , the number of such colorings is bounded by  $O(tw^{2d})$ . Applying  
323 this remark to our algorithm presented in Section 4.1 yields the following result:

324 ► **Theorem 11.** PATH-DIET can be solved in  $O(tw^{2d}n)$ -space and  $O(tw^{4d}n)$ -time.



**Figure 6** Five cases where two vertices are deleted in the same bag with  $d = 1$ . Bags are points in the line, and an interval covering all bags containing  $v$  is drawn for each  $v$  (with an equivalent coloring, see Proposition 5). Cases (a) to (d) can be safely avoided by applying the given transformations. In the example for case (e), however, it is necessary to delete both vertices  $u$  and  $v$  from a central bag. It is sufficient to avoid cases (a) and (b) in order to obtain an XP algorithm for  $d$ .

## 5 Proofs of concept

325

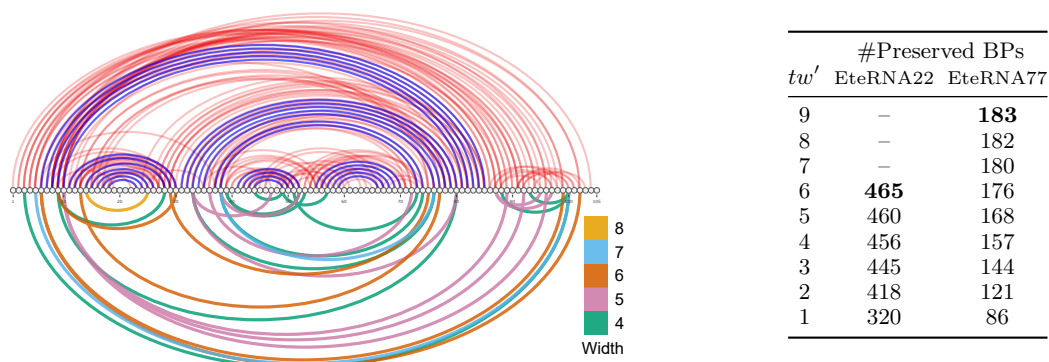
326 We now illustrate the relevance of our approach, and the practicality of our algorithm for  
 327 TREE-DIET, by using it in conjunction with FPT algorithms for three problems in RNA  
 328 bioinformatics. We implemented in C++ the dynamic programming scheme described in  
 329 Theorem 8 and Appendix E. Its main primitives are made available for Python scripting  
 330 through pybind11 [17]. It actually allows to solve a generalized *weighted* version of TREE DIET,  
 331 as explained in Appendix E. This feature allows to favour the conservation of important edges  
 332 (e.g. RNA backbone) during simplification, by assigning them a much larger weight compared  
 333 to other edges. Our implementation is freely available at [https://gitlab.inria.fr/amibio/tree-](https://gitlab.inria.fr/amibio/tree-diet)  
 334 diet.

### 5.1 Memory-parsimonious unbiased sampling of RNA designs

335

336 As a first use case for our simplification algorithm, we strive to ease the sampling phase of a  
 337 recent method, called RNAPond [43], addressing RNA negative design. The method targets  
 338 a set of base pairs  $S$ , representing a secondary structure of length  $n$ , and infers a set  $\mathcal{D}$  of  $m$   
 339 disruptive base pairs (DBPs) that must be avoided. It relies on a  $\Theta(k \cdot (n+m))$  time algorithm  
 340 for sampling  $k$  random sequences (see App. F for details) after a preprocessing in  $\Theta(n \cdot m \cdot 4^{tw})$   
 341 time and  $\Theta(n \cdot 4^{tw})$  space, where  $tw$  is the treewidth of the graph  $G = ([1, n], S \cup \mathcal{D})$ . In  
 342 practice, the preprocessing largely dominates the overall runtime, even for large values of  $k$ ,  
 343 and its large memory consumption represents the main bottleneck.

344 This discrepancy in the complexities/runtimes of the preprocessing and sampling suggests  
 345 an alternative strategy: relaxing the set of constraints to  $(S', \mathcal{D}')$ , with  $(S' \cup \mathcal{D}') \subset (S \cup \mathcal{D})$ ,  
 346 and compensating it through a rejection of sequences violating constraints in  $(S, \mathcal{D}) \setminus (S', \mathcal{D}')$ .  
 347 The relaxed algorithm remains unbiased, while the average-case time complexity of the  
 348 rejection algorithm is in  $\Theta(k \cdot \bar{q} \cdot (n+m))$  time, where  $\bar{q}$  represents the relative increase of the  
 349 partition function ( $\approx$  the sequence space) induced by the relaxation. The preprocessing step  
 350 retains the same complexity, but based on the reduced treewidth  $tw' \leq tw$  of the relaxed  
 351 graph  $G' = ([1, n], S' \cup \mathcal{D}')$ . These complexities enable a tradeoff between the rejection  
 352 (time), and the preprocessing (space), which may be critical to unlock future applications of



■ **Figure 7** (Left) Target secondary structure (blue BPs), full set of disruptive base pairs (DPB; top) inferred by RNAPond on the Eterna77 puzzle, and subsets of DBPs (bottom) cumulatively removed by the tree-diet algorithm to reach prescribed treewidths. (Right) Number of BPs retained by our algorithm, targeting various treewidth values for the EteRNA22 and EteRNA77 puzzles.

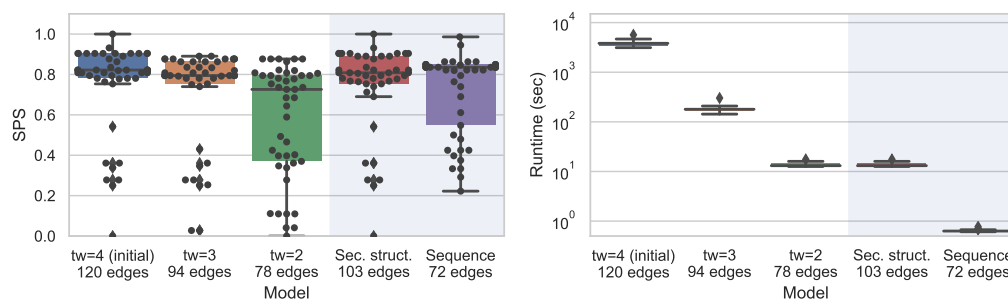
353 RNA design. Indeed, the treewidth can be decreased by removing relatively few base pairs,  
 354 as demonstrated below using our algorithm on pairs inferred for hard design instances.

355 We considered sets of DBPs inferred by RNAPond over two puzzles in the EteRNA  
 356 benchmark. The EteRNA22 puzzle is an empty secondary structure spanning 400 nts, for  
 357 which RNAPond obtains a valid design after inferring 465 DBPs ( $tw = 6$ ). The EteRNA77  
 358 puzzle is 105 nts long, and consists in a collection of helices interspersed with destabilizing  
 359 internal loops. RNAPond failed to produce a solution, and its final set of DBPs consists of  
 360 183 pairs, forming a graph of treewidth  $tw = 9$ . Executing the tree-diet algorithm (Th. 8)  
 361 on both graphs, we obtained a simplified graphs, having lower treewidth while typically  
 362 losing few edges, as illustrated and reported in Figure 7. Remarkably, the treewidth of the  
 363 DBPs inferred for EteRNA22 can be decreased to  $tw' = 5$  by only removing 5 DBPs/edges  
 364 (460/465 retained), and to  $tw' = 4$  by removing 4 further DBPs (456/465). For EteRNA77,  
 365 our algorithm reduces the treewidth from 9 to 6 by only removing 7 DBPs.

366 Rough estimates can be provided for the tradeoff between the rejection and preprocessing  
 367 complexities, by assuming that removing a DBP homogeneously increases the value of  $\mathcal{Z}$  by  
 368 a factor  $\alpha := 16/10$  ( $\#pairs/\#incomp. pairs$ ). The relative increase in partition function is  
 369 then  $\bar{q} \approx \alpha^b$ , when  $b$  base pairs are removed. For EteRNA22, reducing the treewidth by 2  
 370 units ( $6 \rightarrow 4$ ), *i.e.* a 16 fold reduction of the memory and preprocessing time, can be achieved  
 371 by removing 9 DBPs, *i.e.* a 69 fold expected increase in the time of the generation phase. For  
 372 EteRNA77, the same 16 fold ( $tw' = 9 \rightarrow 7$ ) reduction of the preprocessing time/space can  
 373 be achieved through an estimated 4 fold increase of the generation time. A more aggressive  
 374 256 fold memory gain can be achieved at the expense of an estimated 1 152 fold increase  
 375 in generation time. Given the large typical asymmetry in runtimes and implementation  
 376 constants between the computation-heavy preprocessing and, relatively light, generation  
 377 phases, the availability of an algorithm for the TREE-DIET problem provides new options,  
 378 especially to circumvent memory limitations.

## 379 5.2 Structural alignment of complex RNAs

380 Structural homology is often posited within functional families of non-coding RNAs, and is  
 381 foundational to algorithmic methods for multiple RNA alignments [18], considering RNA  
 382 base pairs while aligning distant homologs. In the presence of complex structural features



■ **Figure 8** Impact on alignment quality (SPS; Left) and runtime (Right) of simplified instances for the RNA sequence-structure alignment of the pseudoknotted *c*-di-GMP-II riboswitch. The impact of simplifications on the quality of predicted alignments, using RFAM RF01786 as a reference, appears limited while the runtime improvement is substantial.

383 (pseudoknots, base triplets), the sequence-structure alignment problem becomes hard, yet  
 384 admits XP solutions based on the treewidth of the base pair + backbone graph. In particular,  
 385 Rinaudo *et al* [25] describe a  $\Theta(n \cdot m^{tw+1})$  algorithm for optimally aligning a structured  
 386 RNA of length  $n$  onto a genomic region of length  $m$ . It optimizes an alignment score that  
 387 includes: i) substitution costs for matches/mismatches of individual nucleotides and base  
 388 pairs (including arc-breaking) based on the RIBOSUM matrices [19]; and ii) an affine gap  
 389 cost model [26]. We used the implementation of the Rinaudo *et al* algorithm, implemented  
 390 in the LicoRNA software package [37, 38].

### 391 5.2.1 Impact of treewidth on the structural alignment of a riboswitch

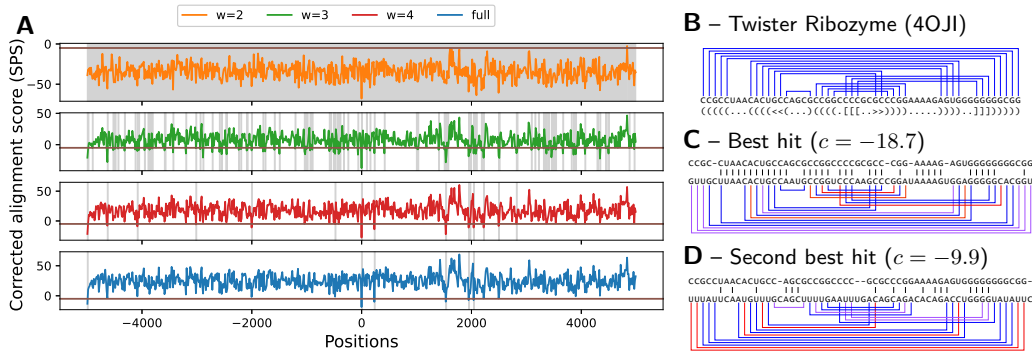
392 In this case study, we used our tree-diet algorithm to modulate the treewidth of complex  
 393 RNA structures, and investigate the effect of the simplification on the quality and runtimes of  
 394 structure-sequence alignments. We considered the Cyclic di-GMP-II riboswitch, a regulatory  
 395 motif found in bacteria that is involved in signal transduction, and undergoes conformational  
 396 change upon binding the second messenger *c*-di-GMP-II [32, 33]. A 2.5Å resolution 3D model  
 397 of the *c*-di-GMP-II riboswitch in *C. acetobutylicum*, proposed by Smith *et al* [30] based on  
 398 X-ray crystallography, was retrieved from the PDB [4] (PDBID: 3Q3Z). We annotated its base  
 399 pairs geometrically using the DSSR method [22]. The canonical base pairs, supplemented  
 400 with the backbone connections, were then accumulated in a graph, for which we heuristically  
 401 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 [18]  
 (RF01786). Finally, we reported the LicoRNA runtime, and computed the Sum of Pairs  
 Score (SPS) [35] as a measure of the accuracy of  $a_{\mathcal{T},w}$  against a reference alignment  $a_w^*$ :

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

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

404 The results, presented in Figure 8, show a limited impact of the simplification on the  
 405 quality of the predicted alignment, as measured by the SPS in comparison with the RFAM



■ **Figure 9** Corrected costs associated with the search for structured homologs of the Twister ribozyme in chromosome 5 of *S. bicolor*, using simplified instances of various treewidth (A). Gray areas represent scores which, upon correction, remain below the cutoff, and have to be considered for further steps of the iterated filtering. Canonical base pairs of the ribozyme (PDBID 4OJI; B), mapped onto to the best hit (C) and second best hit (D) found along the search colored depending on their support in the target sequence (Red: incompatible; Purple: unstable G-U; Blue: stable).

406 alignment. The best average SPS (77.3%) is achieved by the initial model, having treewidth  
 407 of 4, but the average difference with simplified models appears very limited (*e.g.* 76.5% for  
 408  $\mathcal{T}_3$ ), especially when considering the median. Meanwhile, the runtimes mainly depend on the  
 409 treewidth, ranging from 1h for  $\mathcal{T}_4$  to 300ms for  $\mathcal{T}_{1D}$ . Overall,  $\mathcal{T}_{2D}$  seems to represent the  
 410 best compromise between runtime and SPS, although its SPS may be artificially inflated by  
 411 our election of RF01786 as our reference (built from a covariance model, *i.e.* essentially a 2D  
 412 structure). Finally, the difference in number of edges (and induced SPS) between  $\mathcal{T}_{2D}$  and  
 413  $\mathcal{T}_2$ , both having  $tw = 2$ , exemplifies the difference between the TREE-DIET and GRAPH-DIET  
 414 problems, and motivates further work on the latter.

### 415 5.2.2 Exact iterative strategy for the genomic search of ncRNAs

416 In this final case study, we consider an exact filtering strategy to search new occurrences of a  
 417 structured RNA within a given genomic context. In this setting, one attempts to find all  
 418  $\varepsilon$ -admissible (cost  $\leq \varepsilon$ ) occurrences/hits of a structured RNA  $S$  of length  $n$  within a given  
 419 genome of length  $g \gg n$ , broken down in windows of length  $\kappa \cdot n$ ,  $\kappa > 1$ . Classically, one  
 420 would align  $S$  against individual windows, and report those associated with an admissible  
 421 alignment cost. This strategy would have an overall  $\Theta(g \cdot n^{tw+2})$  time complexity, applying  
 422 for instance the algorithm of [25].

423 Our instance simplification framework enables an alternative strategy, that incrementally  
 424 filters out unsuitable windows based on models of increasing granularity. Indeed, for any given  
 425 target sequence, the min alignment cost  $c_\delta$  obtained for a simplified instance of treewidth  
 426  $tw - \delta$  can be corrected (*cf* Appendix G) into a lower bound  $c_\delta^*$  for the min alignment  
 427 cost  $c_0^*$  of the full-treewidth instance  $tw$ . Any window such that  $c_\delta^* > \varepsilon$  thus also obeys  
 428  $c_0^* > \varepsilon$ , and can be safely discarded from the list of putative  $\varepsilon$ -admissible windows, without  
 429 having to perform a full-treewidth alignment. Given the exponential growth of the alignment  
 430 runtime for increasing treewidth values (see Figure 8-right) this strategy is expected to yield  
 431 substantial runtime savings.

432 We used this strategy to search occurrences of the Twister ribozyme (PDBID 4OJI),  
 433 a highly-structured ( $tw = 5$ ) 54nts RNA initially found in *O. sativa* (asian rice) [21]. We  
 434 targeted the *S. bicolor* genome (sorghum), focusing on a 10kb region centered on the 2,485,140

435 position of the 5th chromosome, where an instance of the ribozyme was suspected within  
436 an uncharacterized transcript (LOC110435504). The 4OJI sequence and structure were  
437 extracted from the 3D model as above, and included into a tree decomposition  $\mathcal{T}_5$  (73 edges),  
438 simplified into  $\mathcal{T}_4$  (71 edges),  $\mathcal{T}_3$  (68 edges) and  $\mathcal{T}_2$  (61 edges) using the tree-diet algorithm.

439 We aligned all tree decompositions against all windows of size 58nts using 13nts offset, and  
440 measured the score and runtime of the iterative filtering strategy using a cost cutoff  $\varepsilon = -5$ .  
441 The search recovers the suspected occurrence of twister as its best result (Figure 9.C), but  
442 produced hits (*cf* Figure 9.D) with comparable sequence conservation that could be the  
443 object of further studies. Regarding the filtering strategy, while  $\mathcal{T}_2$  only allows to rule out  
444 3 windows out of 769,  $\mathcal{T}_3$  allows to eliminate an important proportion of putative targets,  
445 retaining only 109 windows, further reduced to 15 windows by  $\mathcal{T}_4$ , 6 of which end up as final  
446 hits for the full model  $\mathcal{T}_5$  (*cf* Figure 9.A). The search remains exact, but greatly reduces the  
447 overall runtime from 24 hours to 34 minutes (42 fold!).

## 448 6 Conclusion and discussion

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

458 **Open questions.** Regarding the parameterized complexity of GRAPH-DIET and TREE-  
459 DIET, some questions remain open (see Table 1): an FPT algorithm for TREE-DIET  
460 (ideally, with  $2^{O(tw)}$  running time), would be the most desirable, if possible satisfying the  
461 backbone constraints. We also aim at settling the parameterized complexity of the GRAPH-  
462 DIET problem, and try to give efficient exact algorithms for this problem (possibly using  
463 some tree-decomposition in input). Finally, we did not include the number of deleted edges  
464 in our multivariate analysis: even though in practice it is more difficult, *a priori*, to guarantee  
465 it has a small value, we expect it can be used to improve the running time in many cases.

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

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



## 480 — References

- 481 1 Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction  
482 with pseudoknots. *Discrete Appl. Math.*, 104(1-3):45–62, 2000. doi:[http://dx.doi.org/10.1016/S0166-218X\(00\)00186-4](http://dx.doi.org/10.1016/S0166-218X(00)00186-4).  
483
- 484 2 Julien Baste, Christophe Paul, Ignasi Sau, and Celine Scornavacca. Efficient FPT algorithms  
485 for (strict) compatibility of unrooted phylogenetic trees. *Bulletin of Mathematical Biology*,  
486 79(4):920–938, feb 2017. doi:10.1007/s11538-017-0260-y.
- 487 3 Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. Hitting minors on bounded treewidth  
488 graphs. i. general upper bounds. *SIAM J. Discret. Math.*, 34(3):1623–1648, 2020. doi:  
489 10.1137/19M1287146.
- 490 4 H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov,  
491 and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28:235–242, January 2000.  
492 doi:10.1093/nar/28.1.235.
- 493 5 Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, and Helene Touzet. Alignments  
494 of RNA structures. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*,  
495 7(2):309–322, apr 2010. doi:10.1109/tcbb.2008.28.
- 496 6 Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth.  
497 *SIAM Journal on computing*, 25(6):1305–1317, 1996.
- 498 7 Laurent Bulteau, Guillaume Fertin, Minghui Jiang, and Irena Rusu. Tractability and approx-  
499 imability of maximal strip recovery. *Theoretical Computer Science*, 440:14–28, 2012.
- 500 8 Laurent Bulteau and Mathias Weller. Parameterized algorithms in bioinformatics: An overview.  
501 *Algorithms*, 12(12):256, dec 2019. doi:10.3390/a12120256.
- 502 9 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary  
503 properties. *Information Processing Letters*, 58(4):171–176, 1996.
- 504 10 Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of  
505 finite graphs. *Information and Computation*, 85(1):12–75, 1990. URL: [https://](https://www.sciencedirect.com/science/article/pii/089054019090043H)  
506 [www.sciencedirect.com/science/article/pii/089054019090043H](https://www.sciencedirect.com/science/article/pii/089054019090043H), doi:[https://doi.org/](https://doi.org/10.1016/0890-5401(90)90043-H)  
507 [10.1016/0890-5401\(90\)90043-H](https://doi.org/10.1016/0890-5401(90)90043-H).
- 508 11 Christophe Crespelle, Pål Grønås Drange, Fedor V Fomin, and Petr A Golovach. A sur-  
509 vey of parameterized algorithms and the complexity of edge modification. *arXiv preprint*  
510 *arXiv:2001.06867*, 2020.
- 511 12 Marek Cygan, Daniel Lokshantov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh.  
512 On the hardness of losing width. In *International Symposium on Parameterized and Exact*  
513 *Computation*, pages 159–168. Springer, 2011.
- 514 13 Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science &  
515 Business Media, 2012.
- 516 14 Ehab S El-Mallah and Charles J Colbourn. The complexity of some edge deletion problems.  
517 *IEEE transactions on circuits and systems*, 35(3):354–362, 1988.
- 518 15 Stefan Hammer, Wei Wang, Sebastian Will, and Yann Ponty. Fixed-parameter tractable  
519 sampling for RNA design with multiple target structures. *BMC Bioinformatics*, 20(1), apr  
520 2019. doi:10.1186/s12859-019-2784-7.
- 521 16 Buhm Han, Banu Dost, Vineet Bafna, and Shaojie Zhang. Structural alignment of  
522 pseudoknotted RNA. *Journal of Computational Biology*, 15(5):489–504, 2008. doi:<http://dx.doi.org/10.1089/cmb.2007.0214>.  
523
- 524 17 Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability  
525 between c++11 and python, 2017. <https://github.com/pybind/pybind11>.
- 526 18 Ioanna Kalvari, Eric P Nawrocki, Nancy Ontiveros-Palacios, Joanna Argasinska, Kevin  
527 Lamkiewicz, Manja Marz, Sam Griffiths-Jones, Claire Toffano-Nioche, Daniel Gautheret,  
528 Zasha Weinberg, Elena Rivas, Sean R Eddy, Robert D Finn, Alex Bateman, and Anton I  
529 Petrov. Rfam 14: expanded coverage of metagenomic, viral and microRNA families. *Nucleic*  
530 *Acids Research*, 49(D1):D192–D200, nov 2020. doi:10.1093/nar/gkaa1047.

- 531 19 Robert J Klein and Sean R Eddy. Rsearch: finding homologs of single structured RNA  
532 sequences. *BMC bioinformatics*, 4(1):44, 2003.
- 533 20 Neocles B Leontis and Eric Westhof. Geometric nomenclature and classification of RNA base  
534 pairs. *RNA*, 7(4):499–512, 2001.
- 535 21 Yijin Liu, Timothy J Wilson, Scott A McPhee, and David MJ Lilley. Crystal structure and  
536 mechanistic investigation of the twister ribozyme. *Nature chemical biology*, 10(9):739–744,  
537 2014.
- 538 22 Xiang-Jun Lu, Harmen J. Bussemaker, and Wilma K. Olson. DSSR: an integrated software  
539 tool for dissecting the spatial structure of RNA. *Nucleic Acids Research*, 43(21):e142–e142, 07  
540 2015. [arXiv:https://academic.oup.com/nar/article-pdf/43/21/e142/17435026/gkv716.pdf](https://academic.oup.com/nar/article-pdf/43/21/e142/17435026/gkv716.pdf), [doi:10.1093/nar/gkv716](https://doi.org/10.1093/nar/gkv716).
- 542 23 R. B. Lyngsø and C. N. S. Pedersen. RNA pseudoknot prediction in energy-based models.  
543 *Journal of Computational Biology*, 7(3-4):409–427, 2000.
- 544 24 Vladimir Reinharz, Antoine Soulé, Eric Westhof, Jérôme Waldispühl, and Alain Denise. Mining  
545 for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network  
546 families. *Nucleic Acids Research*, 46(8):3841–3851, mar 2018. [doi:10.1093/nar/gky197](https://doi.org/10.1093/nar/gky197).
- 547 25 Philippe Rinaudo, Yann Ponty, Dominique Barth, and Alain Denise. Tree decomposition and  
548 parameterized algorithms for RNA structure-sequence alignment including tertiary interactions  
549 and pseudoknots. In *Lecture Notes in Computer Science*, pages 149–164. Springer Berlin  
550 Heidelberg, 2012. [doi:10.1007/978-3-642-33122-0\\_12](https://doi.org/10.1007/978-3-642-33122-0_12).
- 551 26 Elena Rivas and Sean R Eddy. Parameterizing sequence alignment with an explicit evolutionary  
552 model. *BMC bioinformatics*, 16(1):406, 2015.
- 553 27 Toshiki Saitoh, Ryo Yoshinaka, and Hans L. Bodlaender. Fixed-treewidth-efficient algorithms  
554 for edge-deletion to interval graph classes. In *WALCOM: Algorithms and Computation - 15th  
555 International Conference and Workshops, 2021, Proceedings*, volume 12635 of *Lecture Notes in  
556 Computer Science*, pages 142–153. Springer, 2021. [doi:10.1007/978-3-030-68211-8\\_12](https://doi.org/10.1007/978-3-030-68211-8_12).
- 557 28 Roman Sarrazin-Gendron, Hua-Ting Yao, Vladimir Reinharz, Carlos G. Oliver, Yann Ponty,  
558 and Jérôme Waldispühl. Stochastic sampling of structural contexts improves the scalability  
559 and accuracy of RNA 3d module identification. In *Lecture Notes in Computer Science*, pages  
560 186–201. Springer International Publishing, 2020. [doi:10.1007/978-3-030-45257-5\\_12](https://doi.org/10.1007/978-3-030-45257-5_12).
- 561 29 Saad Sheikh, Rolf Backofen, and Yann Ponty. Impact Of The Energy Model On The Complexity  
562 Of RNA Folding With Pseudoknots. In Juha Kärkkäinen and Jens Stoye, editors, *CPM - 23rd  
563 Annual Symposium on Combinatorial Pattern Matching - 2012*, volume 7354 of *Combinatorial  
564 Pattern Matching*, pages 321–333, Helsinki, Finland, July 2012. Juha Kärkkäinen, Springer.  
565 [doi:10.1007/978-3-642-31265-6\\_26](https://doi.org/10.1007/978-3-642-31265-6_26).
- 566 30 Kathryn D. Smith, Carly A. Shanahan, Emily L. Moore, Aline C. Simon, and Scott A. Strobel.  
567 Structural basis of differential ligand recognition by two classes of bis-(3'-5')-cyclic dimeric  
568 guanosine monophosphate-binding riboswitches. *Proceedings of the National Academy of Sci-  
569 ences*, 108(19):7757–7762, 2011. URL: <https://www.pnas.org/content/108/19/7757>, [arXiv:https://www.pnas.org/content/108/19/7757.full.pdf](https://arxiv.org/abs/https://www.pnas.org/content/108/19/7757.full.pdf), [doi:10.1073/pnas.1018857108](https://doi.org/10.1073/pnas.1018857108).
- 571 31 Yinglei Song, Chunmei Liu, Russell Malmberg, Fangfang Pan, and Liming Cai. Tree decom-  
572 position based fast search of RNA structures including pseudoknots in genomes. In *Computational  
573 Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE*, pages 223–234. IEEE,  
574 2005.
- 575 32 N. Sudarsan, E. R. Lee, Z. Weinberg, R. H. Moy, J. N. Kim, K. H. Link, and R. R.  
576 Breaker. Riboswitches in eubacteria sense the second messenger cyclic di-gmp. *Science*,  
577 321(5887):411–413, 2008. URL: <https://science.sciencemag.org/content/321/5887/411>,  
578 [arXiv:https://science.sciencemag.org/content/321/5887/411.full.pdf](https://arxiv.org/abs/https://science.sciencemag.org/content/321/5887/411.full.pdf), [doi:10.1126/  
579 science.1159519](https://doi.org/10.1126/science.1159519).
- 580 33 Rita Tamayo. Cyclic diguanylate riboswitches control bacterial pathogenesis mechanisms.  
581 *PLOS Pathogens*, 15(2):1–7, 02 2019. [doi:10.1371/journal.ppat.1007529](https://doi.org/10.1371/journal.ppat.1007529).

- 582 **34** Jinsong Tan and Louxin Zhang. The consecutive ones submatrix problem for sparse matrices.  
583 *Algorithmica*, 48(3):287–299, 2007.
- 584 **35** J D Thompson, F Plewniak, and O Poch. BALiBASE: a benchmark alignment data-  
585 base for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1):87–88, 01  
586 1999. arXiv:[https://academic.oup.com/bioinformatics/article-pdf/15/1/87/9731974/](https://academic.oup.com/bioinformatics/article-pdf/15/1/87/9731974/150087.pdf)  
587 [150087.pdf](https://academic.oup.com/bioinformatics/article-pdf/15/1/87/9731974/150087.pdf), doi:10.1093/bioinformatics/15.1.87.
- 588 **36** Jelena Vucinic, David Simoncini, Manon Ruffini, Sophie Barbe, and Thomas Schiex. Pos-  
589 itive multistate protein design. *Bioinformatics*, 36(1):122–130, jun 2019. doi:10.1093/  
590 [bioinformatics/btz497](https://academic.oup.com/bioinformatics/article-pdf/36/1/122/2511974/bioinformatics/btz497).
- 591 **37** Wei Wang. *Practical sequence-structure alignment of RNAs with pseudoknots*. PhD thesis,  
592 Université Paris-Saclay, School of Computer Science, 2017.
- 593 **38** Wei Wang, Alain Denise, and Yann Ponty. Licorna: alignment of complex rnas v1.0, 2017.  
594 URL: <https://licorna.lri.fr>.
- 595 **39** M. S. Waterman. Secondary structure of single stranded nucleic acids. *Advances in Mathematics*  
596 *Supplementary Studies*, 1(1):167–212, 1978.
- 597 **40** Mathias Weller, Annie Chateau, and Rodolphe Giroudeau. Exact approaches for scaffolding.  
598 *BMC Bioinformatics*, 16(S14), oct 2015. doi:10.1186/1471-2105-16-s14-s2.
- 599 **41** A. Xayaphoummine, T. Bucher, F. Thalmann, and H. Isambert. Prediction and statistics of  
600 pseudoknots in RNA structures using exactly clustered stochastic simulations. *Proc. Natl.*  
601 *Acad. Sci. U. S. A.*, 100(26):15310–15315, 2003.
- 602 **42** Jinbo Xu. Rapid protein side-chain packing via tree decomposition. In *Lecture Notes in*  
603 *Computer Science*, pages 423–439. Springer Berlin Heidelberg, 2005. doi:10.1007/11415770\_  
604 32.
- 605 **43** Hua-Ting Yao, Jérôme Waldispühl, Yann Ponty, and Sebastian Will. Taming Disruptive Base  
606 Pairs to Reconcile Positive and Negative Structural Design of RNA. In *RECOMB 2021 - 25th*  
607 *international conference on research in computational molecular biology*, Padova, France, April  
608 2021.

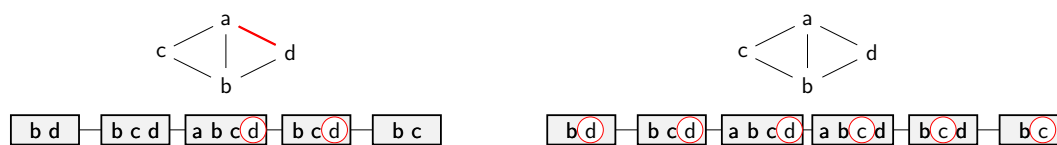


Figure 10 Left: A graph and a path-decomposition whose optimal 1-tree diet loses an edge (ad). However, duplicating the bag abcd (right) yields a tree-decomposition with a lossless 1-tree diet.

609 **A Editing Trees before the Diet**

610 The goal of TREE-DIET is to fix the tree-decomposition structure in order to help achieve  
 611 faster algorithms: finding the best tree-decomposition to give in input would a priori be  
 612 a difficult task. However, given a tree-decomposition  $\mathcal{T}$ , some editions can be performed  
 613 to obtain a tree-decomposition  $\mathcal{T}'$ , that (1) take linear time and (2) may only improve the  
 614 tree-diet solution. In particular, duplicating nodes may only improve the solution (and may  
 615 strictly improve it in some cases, see Figure 10). The same applies to common operations  
 616 to obtain *nice* tree-decompositions (i.e. binary with at most 1 vertex difference between  
 617 adjacent bags), except joining identical nodes (by the example of Figure 10 again). So overall  
 618 it is safe to assume that the input tree-decomposition satisfies the properties of  $\mathcal{T}'$  such as  
 619 being binary with small bag differences.

620 **B Postponed Proofs in Section 3.2**

621 **Proof of Theorem 1.** We consider the decision version of GRAPH-DIET where a bound  $k$   
 622 on the number of deleted edges is given. We build a Turing reduction from TREewidth:  
 623 more precisely, assuming an oracle for GRAPH-DIET with  $d = 1$  is available, we build a  
 624 polynomial-time algorithm to compute the treewidth of a graph  $G$ . This is achieved by  
 625 computing GRAPH-DIET( $G, tw, d = 1, k = 0$ ) for decreasing values of  $tw$  (starting with  
 626  $tw = |V|$ ): the first value of  $tw$  for which this call returns no solution is the treewidth of  
 627  $G$ . Note that this is not a *many-one* reduction, since several calls to GRAPH-DIET may be  
 628 necessary (so this does not precisely qualify as an NP-hardness reduction, even though a  
 629 polynomial-time algorithm for GRAPH-DIET( $G, tw, d = 1, k = 0$ ) would imply P=NP). ◀

630 **Proof of Theorem 4.** By reduction from CLIQUE. Given a  $\delta$ -regular graph  $G$  with  $n$  vertices  
 631 and  $m$  edges and an integer  $k$ , consider the trivial tree-decomposition  $\mathcal{T}$  of  $G$  with a single  
 632 bag containing all vertices of  $G$  (it has width  $n - 1$ ). Then  $(\mathcal{T}, G)$  has a  $k$ -tree diet losing  
 633  $\delta k - \binom{k}{2}$  edges if and only if  $G$  has a  $k$ -clique. Indeed, such a tree diet  $\mathcal{T}'$  would remove a set  
 634  $X$  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  
 635  $G$ . ▶

636 **C Postponed proofs of dynamic programming scheme**

637 **Proof of Lemma 7.** Given, in a tree-decomposition, a bag  $P$  colored with  $f$ , with a child  $X$   
 638 colored with  $h$ , a more precise definition for  $count(f, h)$  is:

639 
$$count(f, h) = \left\{ (u, v) \in E \mid \begin{array}{l} h(u) = h(v) = \text{green} \text{ and} \\ (u \notin f^{-1}(*) \text{ or } f(u) \neq g \text{ or } v \notin f^{-1}(*) \text{ or } f(v) \neq g) \end{array} \right\}$$

640

641 Now, given a realizable edge  $(u, v)$ , in a tree-decomposition  $\mathcal{T}$  colored with  $\mathcal{C}$ , the set of  
 642 bags in which both  $u$  and  $v$  are green forms a connected sub-tree of  $\mathcal{T}$ . This sub-tree has a

643 root, or *lowest common ancestor*, that we denote  $R_{(u,v)}$ . Since we assumed  $\mathcal{T}$  to be rooted  
 644 at an empty bag,  $R_{(u,v)}$  is not the root of  $\mathcal{T}$ , and has a parent. We call this parent  $P_{(u,v)}$ .  
 645 Clearly,  $(u, v)$  belongs to the “count set” associated to the edge  $(P_{(u,v)}) \rightarrow (R_{(u,v)})$  of  $\mathcal{T}$ ,  
 646 while for any other edge  $X \rightarrow Y$  of  $\mathcal{T}$ , the colors of  $u$  and  $v$  cannot verify the conditions to  
 647 belong to the associated “count set”. ◀

648 **Proof of Theorem 6.**

649  $\boxed{\leq}$  By definition,  $c(X_i, f) = \max_{\mathcal{C} \in \mathcal{C}(\mathcal{T}_i, f)} |RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)|$  is the maximum number of realiz-  
 650 able edges in the sub-tree-decomposition rooted at  $X_i$ , such that all green-green occurrences  
 651 of the edge occur strictly below  $X_i$ , and under the constraint that  $f$  colors  $X_i$ . Let  $\mathcal{C}$  be a  
 652 coloring for  $\mathcal{T}_i$  realizing the optimum  $c(X_i, f)$ . Its restrictions to  $Y_1 \dots Y_{\Delta}$  yield colorings  
 653  $f'_1 \dots f'_{\Delta}$ . Likewise, its restrictions to the sub-tree-decompositions  $\mathcal{T}'_1 \dots \mathcal{T}'_{\Delta}$  rooted at  
 654  $Y_1 \dots Y_{\Delta}$  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  
 655 than the optimal, so  $\forall j, |RE_{\downarrow}(\mathcal{T}'_j, \mathcal{C}'_j, G)| \leq c(Y_j, f'_j)$   
 656 Let  $(u, v)$  be an edge of  $RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)$ . Per Lemma 7, either  $(u, v) \in \text{count}(f, f'_j)$  for some  
 657  $j$  (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  $\exists j$   
 658 such that  $(u, v) \in RE_{\downarrow}(\mathcal{T}'_j, \mathcal{C}'_j, G)$ . Therefore:  
 659

$$\begin{aligned} 660 \quad c(X_i, f) = |RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)| &= \sum_{1 \leq j \leq \Delta} [|RE_{\downarrow}(\mathcal{T}'_j, \mathcal{C}'_j, G)| + \text{count}(f, f'_j)] \\ 661 &\leq \sum_{1 \leq j \leq \Delta} (c(Y_j, f'_j) + \text{count}(f, f'_j)) \\ 662 \end{aligned}$$

663 and, a fortiori

$$664 \quad c(X_i, f) \leq \max_{m: f^{-1}(o) \rightarrow [1 \dots \Delta]} \sum_{1 \leq j \leq \Delta} \max_{f'_j \in \text{comp}(Y_j, f, m)} (c(Y_j, f'_j) + \text{count}(f, f'_j))$$

666  $\boxed{\geq}$  Conversely, given  $f$ , let  $m$  be an assignation map for orange vertices and  $f'_1 \dots f'_{\Delta}$   
 667 colorings of  $Y_1 \dots Y_{\Delta}$  compatible with  $f$  and  $m$ , and let  $\mathcal{C}'_1 \dots \mathcal{C}'_{\Delta}$  be colorings of  $\mathcal{T}'_1 \dots \mathcal{T}'_{\Delta}$   
 668 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  $\mathcal{C}$   
 669 for  $\mathcal{T}_i$ , the sub-tree-decomposition rooted at  $X_i$ , which can not be better than optimal  
 670 ( $|RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)| \leq c(X_i, f)$ ). As before, an edge  $(u, v)$  either belongs to  $\cup_j \text{count}(f, f'_j)$  or  
 671 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:

$$\begin{aligned} 672 \quad \sum_{1 \leq j \leq \Delta} (c(Y_j, f'_j) + \text{count}(f, f'_j)) &= \sum_{1 \leq j \leq \Delta} (|RE_{\downarrow}(\mathcal{T}'_j, \mathcal{C}'_j, G)| + \text{count}(f, f'_j)) \\ 673 &= |RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)| \\ 674 &\leq c(X_i, f) \\ 675 \end{aligned}$$

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

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

677 which concludes the proof. ◀

679 **Proof of Theorem 8.** Given the sub-problems and  $c(X_i, f)$ -table definitions, with  $R$  the  
 680 (empty) root of the tree-decomposition,  $c(R, \emptyset)$  is indeed the maximum possible number of

681 realizable edges when imposing a  $(tw - tw')$ -diet to  $\mathcal{T}$ . The recurrence relation of Theorem 6  
 682 therefore for a dynamic programming approach, over the tree-decomposition  $\mathcal{T}$  following  
 683 leaf-to-root order, for the problem.

684 The number of entries to the table is  $O(3^{tw}n)$ , given that a bag  $X$  may be colored in  
 685  $3^{|X|}$  ways, and that the maximum size of  $X$  is  $tw + 1$ . For a given entry  $X_i$ , one must  
 686 first enumerate all possible choices of  $m : f^{-1}(o) \rightarrow [1 \dots \Delta]$ , map assigning one child of  $X_i$   
 687 to each orange vertex in  $X_i$ . There are  $O(\Delta^{tw+1})$  possibilities for  $m$  in the worst case, as  
 688  $|f^{-1}(o)| \leq tw + 1$ . Then, for each child  $Y_j$ , one must enumerate all possible colorings  $f'_j$   
 689 compatible with  $f$ . Possibilities for  $f'_j(u)$  depend on the color by  $f$ :

- 690 ■ if  $u \notin X_i \rightarrow f'_j(u) = o$  or  $g$
- 691 ■ if  $f(u) = g \rightarrow f'_j(u) = g$  or  $r$
- 692 ■ if  $f(u) = o \rightarrow f'_j(u) = o$  or  $g$  if  $m[u] = j$  or  $f'_j(u) = r$  otherwise.
- 693 ■ if  $f(u) = r \rightarrow f'_j(u) = r$

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

## 698 **D** Postponed proofs of path decomposition case

699 **Proof of Proposition 10.** Consider such a coloring  $\mathcal{C}$  with a maximal number of green  
 700 vertices. We show that it is  $d$ -simple. Assume the path-decomposition  $\mathcal{T}$  is rooted in bag  
 701  $X_1$  and each  $X_i$  is the parent of  $X_{i+1}$ . Pick  $i$  to be the smallest index so that at least  $d + 1$   
 702 vertices in  $X_i$  are colored red by  $\mathcal{C}$ , assume any such  $i$  exists. Then one of these vertices,  
 703 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  
 704 or green in  $X_{i-1}$ ). Consider  $\mathcal{C}'$  obtained by  $\mathcal{C}$  and coloring  $u$  green in  $X_i$ . Then  $\mathcal{C}'$  satisfies  
 705 local rules R1 through R4 (a green vertex may be absent, green or orange in the parent bag,  
 706 and a red vertex may be green in the parent bag). Furthermore, it is  $d$ -diet-valid since it  
 707 still removes at least  $d$  (red) vertices in  $X_i$ . Overall  $\mathcal{C}'$  is another  $d$ -diet-valid coloring with  
 708 more green vertices: a contradiction, so no such  $i$  exist (and no bag has  $d + 1$  red vertices).  
 709 The same argument works symmetrically for orange vertices. Overall,  $\mathcal{C}$  is  $d$ -simple. ◀

## 710 **E** Pseudo-code

711 Algorithm 1 and 2 present a pseudo-code of our dynamic programming algorithm for TREE  
 712 DIET, with a memoization approach. The C++/pybind11 [17] implementation is available  
 713 at <https://gitlab.inria.fr/amibio/tree-diet>.

714 Note that the implementation allows to solve a more general *weighted* version of TREE  
 715 DIET, where each edge is given a weight, and the objective is to find a  $(tw - tw')$ -diet of the  
 716 input tree decomposition preserving a set of edges of maximum total weight.

717 In the context of RNA applications, this feature allows to favour as much as possible  
 718 preservation of the backbone of RNA molecules, i.e. edges between consecutive nucleotides  
 719 along the string, by assigning them a weight greater than the number of non-backbone edges.

720 Edge weights are passed to the function in the form of a dictionary/map  $W$  associating a  
 721 real weight to each edge. Within Algorithms 1 and 2, the only place where it is taken into  
 722 account is the the *count* function, which computes the weight of edges accounted for by the  
 723 bag that is currently visited.

■ **Algorithm 1** Dynamic programming algorithm for TREE-DIET.

```

Input      : Tree-decomposition  $\mathcal{T}$ , graph  $G$ , target width  $tw'$ , edge weights  $W$ 
Output    : Maximum total weight of a set of realizable/non-lost edges in a
               $(tw - tw')$ -diet of  $\mathcal{T}$ 
Side-Product: A filled table  $c[X_i, f]$ ,  $\forall X_i$  bag and  $f$  coloring of  $X_i$ 

1 Function optim_num_real_edges( $X_i, f, G, tw', W$ ):
2   if  $c[X_i, f]$  already computed then return  $c[X_i, f]$ ; ;
3   if  $|f^{-1}(o) \cup f^{-1}(r)| \leq (|X_i| - tw' - 1)$  then
4     //not enough removals.;
5      $c[X_i, f] = -\infty$ ;
6     return  $c[X_i, f]$ ;
7   end
8   if  $X_i == \text{leaf}$  then
9      $c[X_i, f] = 0$ ;
10    return  $c[X_i, f]$ ;
11  end
12  int ans =  $-\infty$ ;
13  for  $m \in \text{orange\_maps}(X_i, f)$  do
14    int ans_m = 0;
15    for  $Y_j \in X_i.\text{children}$  do
16      int ans_j =  $-\infty$ ;
17      for  $f'_j \in \text{compatible}(f, m, X_i, Y_j)$  do
18        int val = 0;
19         $val += \text{count}(f, f'_j, W)$ ;
20         $val += \text{optim\_num\_real\_edges}(Y_j, f'_j, G, tw')$ ;
21        if  $val \geq \text{ans}_j$  then
22          ans_j = val;
23        end
24      end
25       $\text{ans}_m += \text{ans}_j$ ;
26    end
27    if  $\text{ans}_m \geq \text{ans}$  then
28      ans = ans_m;
29    end
30  end
31   $c[X_i, f] = \text{ans}$ 
32  return  $c[X_i, f]$ ;
33 end

```

■ **Algorithm 2** Backtracking procedure for TREE-DIET.

```

Input      : Tree-decomposition  $\mathcal{T}$ , graph  $G$ , target width  $tw'$ , table  $c$ , edge
              weights  $W$ 
Output    : Optimal  $(tw - tw')$ -diet-valid coloring for  $\mathcal{T}$ 

1 Function optim_coloring( $X_i, f, G, tw', c$ ):
2   if  $X_i == leaf$  then
3     | return  $\emptyset$ ;
4   end
5   coloring  $\mathcal{C} = \emptyset$ ;
6   for  $m \in orange\_maps(X_i, f)$  do
7     | int ans_m = 0;
8     | coloring best_fjs = [];
9     | for  $Y_j \in X_i.children$  do
10    |   | int best_valj =  $-\infty$ ;
11    |   | int best_fj =  $\emptyset$ ;
12    |   | for  $f'_j \in compatible(f, m, X_i, Y_j)$  do
13    |   |   | int val = 0;
14    |   |   | val += count( $f, f'_j, W$ );
15    |   |   | val += c[ $Y_j, f'_j$ ];
16    |   |   | if val  $\geq best\_valj$  then
17    |   |   |   | best_valj = val;
18    |   |   |   | best_fj =  $f'_j$ ;
19    |   |   | end
20    |   | end
21    |   | ans_m += best_valj;
22    |   | best_fjs.add(best_fj);
23    |   | end
24    |   | if ans_m == c[ $X_i, f$ ] then
25    |   |   | C += [ $f'_j$  for  $f'_j$  in best_fjs];
26    |   |   | C += [optim_coloring( $Y_j, f'_j, G, tw', c$ ) for  $f'_j$  in best_fjs];
27    |   |   | break; // break loop over  $m$ 
28    |   | end
29  end
30  return  $\mathcal{C}$ ;
31 end

```



## 724 **F** Correctness of the rejection-based sampling of candidate RNA 725 designs

726 A recent method for RNA design, called RNAPond [43], implements a sampling approach  
727 to tackle the inverse folding of RNA. Targeting a secondary structure  $S$  of length  $n$ , it  
728 performs a Boltzmann-weighted sampling of sequences and, at each iteration, identifies  
729 Disruptive Base Pairs (DBPs) that are not in  $S$ , yet are recurrent in the Boltzmann ensemble  
730 of generated sequences. Those base pairs are then added to a set  $\mathcal{D}$  of DBPs, and excluded  
731 in subsequent generations through an assignment of non-binding pairs of nucleotides, outside  
732 of  $\mathcal{B} := \{(G, C), (C, G), (A, U), (U, A), (G, U), (U, G)\}$ .

733 At the core of the method, one finds a random generation algorithm which takes as input  
734 a secondary structure  $S$  and a set  $\mathcal{D}$  of DBPs. The algorithm generates from the set  $\mathcal{W}_{S,\mathcal{D}}$   
735 of sequences  $w \in \{A, C, G, U\}^n$  which are: i) compatible with all  $(i, j) \in S$ , *i.e.*  $(w_i, w_j) \in \mathcal{B}$ ;  
736 and ii) incompatible with all  $(k, l) \in \mathcal{D}$ , *i.e.*  $(w_k, w_l) \notin \mathcal{B}$ . The algorithm then enforces a  
737 (dual) Boltzmann distribution over the sequences in  $\mathcal{W}_{S,\mathcal{D}}$ :

$$738 \quad \forall w \in \mathcal{W}_{S,\mathcal{D}} : \mathbb{P}(w \mid \mathcal{D}, S) = \frac{e^{-\beta \cdot E_{w,S}}}{\mathcal{Z}_{S,\mathcal{D}}} \quad \text{with} \quad \mathcal{Z}_{S,\mathcal{D}} := \sum_{w' \in \mathcal{W}_{S,\mathcal{D}}} e^{-\beta \cdot E_{w',S}} \quad (1)$$

739 where  $\beta > 0$  is an arbitrary constant akin to a temperature. Yao *et al* describe an algorithm  
740 which generates  $k$  sequences in  $\Theta(k(n + |\mathcal{D}|))$  time, after a preprocessing in  $\Theta(n \cdot |\mathcal{D}| \cdot 4^{tw})$   
741 time and  $\Theta(n \cdot 4^{tw})$  space, where  $tw$  is the treewidth of the graph having edges in  $S \cup \mathcal{D}$ .

The discrepancy in the preprocessing and sampling complexities suggests an alternative  
strategy, utilizing rejection on top of a relaxed sampling. Namely, we consider a rejection  
algorithm, which starts from a relaxation  $(S', \mathcal{D}')$  of the initial constraints  $(S' \cup \mathcal{D}' \subset S \cup \mathcal{D})$ ,  
and iterates Yao *et al*'s algorithm to generate sequences in  $\mathcal{W}_{S',\mathcal{D}'} \supset \mathcal{W}_{S,\mathcal{D}}$ , rejecting those  
outside of  $\mathcal{W}_{S,\mathcal{D}}$ , until  $k$  suitable ones are obtained. The rejection algorithm generates a  
given sequence  $w \in \mathcal{W}_{S,\mathcal{D}}$  on its first attempt with probability  $p := e^{-\beta \cdot E_{w,S}} / \mathcal{Z}_{S',\mathcal{D}'}$  and,  
more generally, after  $r$  rejections with probability  $(1 - q)^r p$  with  $q := \mathcal{Z}_{S,\mathcal{D}} / \mathcal{Z}_{S',\mathcal{D}'}$ . The  
overall probability of emitting  $w$  is thus

$$p \cdot \sum_{r \geq 0} (1 - q)^r = \frac{p}{q} = \frac{e^{-\beta \cdot E_{w,S}}}{\mathcal{Z}_{S,\mathcal{D}}} = \mathbb{P}(w \mid \mathcal{D}, S).$$

742 In other words, our relaxed generator coupled with the rejection step, represents an unbiased  
743 algorithm for the Boltzmann distribution of Eq. (1) over  $\mathcal{W}_{S,\mathcal{D}}$ .

744 Meanwhile, the average-case complexity can be impacted by the strategy. Indeed,  
745 the relaxed instance  $(S', \mathcal{D}')$  can accelerate the preprocessing due to a reduced treewidth  
746  $tw' \leq tw$ . The rejection step only increases the expected number of generations by a factor  
747  $\bar{q} := \mathcal{Z}_{S',\mathcal{D}'} / \mathcal{Z}_{S,\mathcal{D}}$ , representing the inflation of the sequence space, induced by the relaxation  
748 of the constraints. Overall, the average-case time complexity of the rejection algorithm is in  
749  $\Theta(n \cdot |\mathcal{D}'| \cdot 4^{tw'} + k \cdot \bar{q} \cdot (n + |\mathcal{D}'|))$  time and  $\Theta(n \cdot 4^{tw'})$  space. This space improvement is notable  
750 when  $tw' < tw$ , and could be key for the practical applicability of the method, especially  
751 given that memory represents the bottleneck of most treewidth-based DP algorithms.

## 752 **G** Lower bound for the min. alignment cost from simplified models

753 Here, we justify the filtering strategy described in Section 5.2.2. Namely, we formally prove  
754 that, given a structured RNA  $S$  and a targeted genomic region  $w$ , a lower bound for the

755 minimal alignment cost of  $S$  and  $w$  can be obtained from the minimal alignment cost of  
 756 some  $S' \subseteq S$  and  $w$ . If this lower bound for  $S' \subseteq S$  is higher than the specified cutoff  $\varepsilon$ , then  
 757 there is no need to align  $w$  to  $S$  the full model, as the resulting cost is guaranteed to stay  
 758 above the selection cutoff  $\varepsilon$ .

759 Let  $S$  be an arc-annotated sequence of length  $m$  ( $S_i$  denotes the  $i$ th character of  $S$ ),  $w$  be  
 760 a target (flat) sequence of length  $m$ , and  $\mu : [1, n] \rightarrow [1, m] \cup \{\perp\}$  represents an alignment<sup>2</sup>.  
 761 We consider the following cost function, adapted from [25], which quantifies the quality of an  
 762 alignment  $\mu$  for  $S$  and  $w$ :

$$763 \quad C(S, w, \mu) = \sum_{\substack{i \text{ unpaired in } S, \\ k := \mu_i}} \gamma(S_i, w_k) + \sum_{\substack{(i,j) \in S, \\ (k,l) := (\mu_i, \mu_j)}} \phi(S_i, S_j, w_k, w_l) \\ 764 \quad + \sum_{g \in \text{gaps}(S)} \lambda_q(g) + \sum_{g \in \text{gaps}(w)} \lambda_T(g) \\ 765$$

766 where

- 767 ■  $\gamma(a, b)$  returns the *substitution cost* which penalizes (mismatches) or rewards (matches)  
 768 the substitution of  $a$  into  $b$  (set to 0 and handled in gaps if  $b = \perp$ );
- 769 ■  $\phi(a, b, c, d)$  return a *base pair substitution cost*, penalizing (arc breaking) or rewarding  
 770 (conservation or compensatory mutations) the transformation of nucleotides  $(a, b)$  into  
 771 nucleotides/gaps  $(c, d)$  (set to 0 and handled in gaps if  $(c, d) = (\perp, \perp)$ );
- 772 ■  $\lambda_S$  and  $\lambda_T$  penalize gaps introduced by  $\mu$  respectively in  $S$  and  $w$  (affine cost model).

Given this definition, consider a simplified model  $S' \subset S$ , associated with a minimal cost

$$c' := \min_{\mu} C(S, w, \mu)$$

773 and denote by  $c^*$  the minimal cost of the full model  $S$ , we have the following inequality.

► **Proposition 12.**

$$774 \quad c' - \sum_{\substack{i \text{ unpaired in } S', \\ \text{paired in } S}} \max_b \gamma(S_i, b) + \sum_{(i,j) \in S \setminus S'} \min_{a,b} \phi(S_i, S_j, a, b) \leq c^* \quad (2)$$

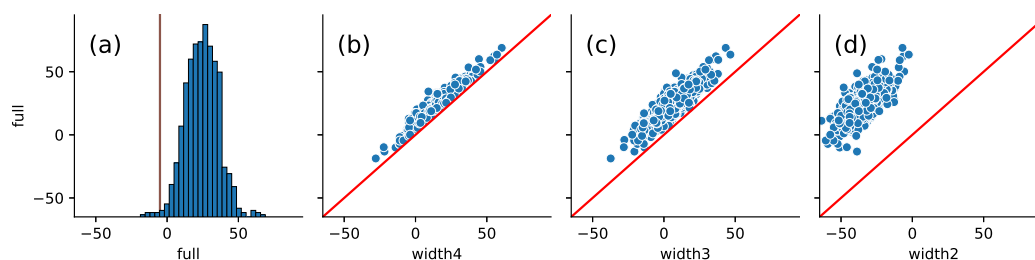
775 **Proof.** For any alignment, we have, per the definition of  $C(S, w, \mu)$ :

$$776 \quad C(S, w, \mu) = C(S', w, \mu) - \sum_{\substack{i \text{ unpaired in } S', \\ \text{paired in } S, \\ \text{and } k := \mu_i}} \gamma(S_i, w_k) + \sum_{\substack{(i,j) \in S \setminus S' \\ \text{s.t. } (k,l) := (\mu_i, \mu_j)}} \phi(S_i, S_j, w_k, w_l). \\ 777$$

778 Minimizing over all alignment  $\mu$ , one obtains

$$779 \quad \min_{\mu} C(S, w, \mu) = \min_{\mu} C(S', w, \mu) - \sum_{\substack{i \text{ unpaired in } S', \\ \text{paired in } S, \\ \text{and } k := \mu_i}} \gamma(S_i, w_k) + \sum_{\substack{(i,j) \in S \setminus S' \\ \text{s.t. } (k,l) := (\mu_i, \mu_j)}} \phi(S_i, S_j, w_k, w_l). \\ 780$$

<sup>2</sup> An alignment  $\mu$  is subject to further constraints, notably including some restricted form of monotonicity, when represented as a function. However, those constraints are reasonably intuitive and we omit them in this discussion for the sake of simplicity.



■ **Figure 11** (a) Histogram of alignment scores obtained by aligning the full structure ( $tw = 5$ ) model of the Twister ribozyme (pdb-id: 4OJI) with  $\kappa \cdot n$ -sized windows in a 10kb region of the 5<sup>th</sup> chromosome of *S. bicolor*. A vertical line is positioned at the  $\epsilon$  threshold. (b;c;d) Corrected alignment scores obtained for reduced-treewidth models for each window, plotted against the corresponding score of the full model. The corrected alignment score indeed acts as a lower bound to the full-model score (points above the  $y = x$  red line), allowing a iterative filtering strategy.

781 Independently minimizing each term of the right-hand-side, we obtain a first lower bound

$$782 \quad c^* \geq c' - \max_{\mu} \sum_{\substack{i \text{ unpaired in } S', \\ \text{paired in } S, \\ \text{and } k := \mu_i}} \gamma(S_i, w_k) + \min_{\mu} \sum_{\substack{(i,j) \in S \setminus S' \\ \text{s.t. } (k,l) := (\mu_i, \mu_j)}} \phi(S_i, S_j, w_k, w_l).$$

783  
784 further coarsened by an independent optimization of the elements in the sums

$$785 \quad c^* \geq c' - \sum_{\substack{i \text{ unpaired in } S', \\ \text{paired in } S}} \max_{\mu} \gamma(S_i, w_k) + \sum_{(i,j) \in S \setminus S'} \min_{\mu} \phi(S_i, S_j, w_k, w_l)$$

$$786 \quad = c' - \sum_{\substack{i \text{ unpaired in } S', \\ \text{paired in } S}} \max_a \gamma(S_i, a) + \sum_{(i,j) \in S \setminus S'} \min_{a,b} \phi(S_i, S_j, a, b).$$

787  
788 where the last line is obtained by considering the worst-case contributors to nucleotides and  
789 base pairs substitutions. Importantly, the right-hand side no longer depends on  $\mu$  any more,  
790 and can be used to easily computed a corrected score/lower bound. ◀

791 The corrected expression, shown in the left hand side of Equation (2) allows, when lower  
792 than a cutoff  $\epsilon$ , to safely discard  $w$  as a potential hit for the full model  $S$ . This corrected  
793 score score is plotted in Figure 9A, allowing for a gradual reduction of the search space  
794 for  $\epsilon$ -admissible hits. We show in Figure 11 the corrected scores obtained for simplified  
795 structures  $S'$  of various treewidths, plotted against the scores of the full target structure.