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

Bertrand Marchand, Yann Ponty, Laurent Bulteau

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[1]
LIX CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
yann.ponty@lix.polytechnique.fr

Laurent Bulteau[1]
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 → Dynamic programming; Theory of computation → Parameterized complexity and exact algorithms; Applied computing → 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

[1] To whom correspondence should be addressed

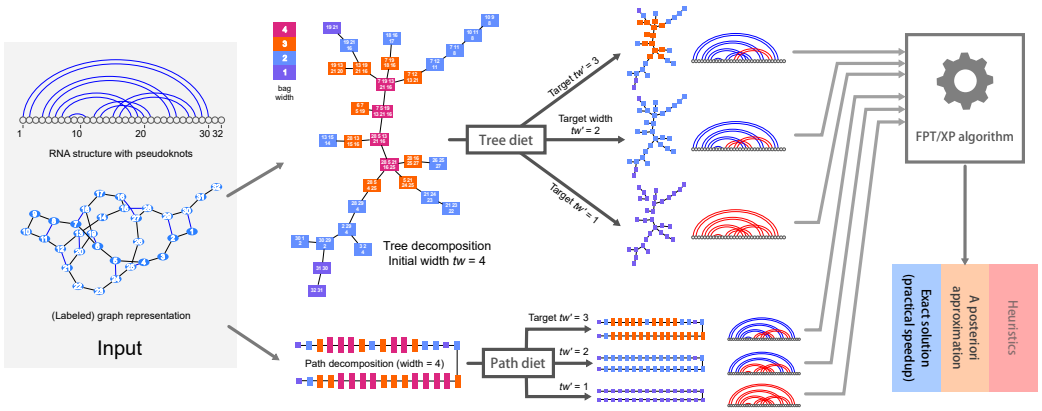


Figure 1 General description of our approach and rationale. Starting from a structured instance, *e.g.* an RNA structure with pseudoknots, our tree diet/path diet algorithms extract simplified tree/path decompositions, having prescribed target width tw'. Those can be used within existing parameterized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

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

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

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

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

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

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

2 Statement of the problem(s) and results

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

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

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

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

We define Binary-Tree-Diet and Path-Diet analogously, where $\mathcal{T}$ is restricted to be a binary tree (respectively, a path). An example is given in Figure 2.

Parameterized Complexity in a Nutshell

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

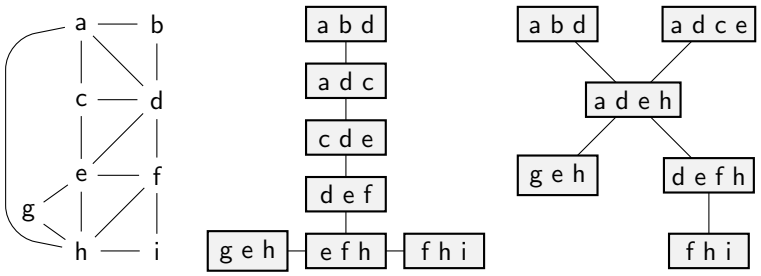


Figure 2 Illustrations for the Graph-Diet and Tree-Diet problems. Given a graph G on the left (treewidth 3), an optimal solution for Graph-Diet yields the tree-decomposition in the middle (width 2, edge ah is lost). On the other hand, any 1-tree diet for the tree-decomposition on the right loses at least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

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

2.1 Our results

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

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

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

3 Algorithmic Limits: Parameterized Complexity Considerations

3.1 The intricate Graph-Diet Problem

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

<table>
<tr><th>Parameter
Problem</th><th colspan="2">Source treewidth
tw</th><th>Target treewidth
tw'</th><th>Difference
$d = tw - tw'$</th></tr>
<tr><td>Graph-Diet</td><td colspan="2">open</td><td>Para-NP-hard
$tw' = 2$
EDP(K_4) [14]</td><td>Para-NP-hard*
$d = 1$
Theorem 1</td></tr>
<tr><td>Tree-Diet</td><td>XP
$O^*((6\Delta)^{tw})$
Theorem 8</td><td>FPT open</td><td rowspan="3">Para-NP-hard
$tw' = 1$
Theorem 2</td><td>Para-NP-hard
$d = 1$
Theorem 3</td></tr>
<tr><td>Binary-Tree-Diet</td><td colspan="2" rowspan="2">FPT
$O^*(12^{tw})$
Theorem 8</td><td rowspan="2">W[1]-hard
Theorem 4</td><td>XP open</td></tr>
<tr><td>Path-Diet</td><td>XP
$O^*(tw^d)$
Theorem 11</td></tr>
</table>

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

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

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

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

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

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

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

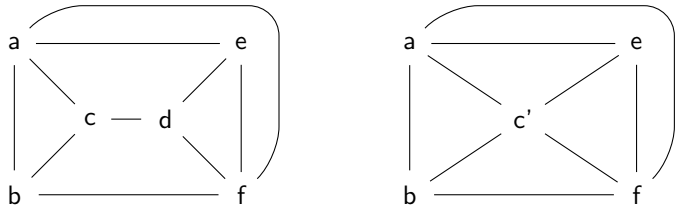


Figure 3 A graph G (left) with treewidth 3. Deleting edge cd gives treewidth 2. However, if one contracts edge cd, then the resulting graph (right) has treewidth 3, and deleting any single edge does not decrease the treewidth. This example shows that the graph family Treewidth 2+1e is not minor-closed.

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

▶ **Conjecture 1.** Graph-Diet *is* FPT *for the source treewidth parameter (tw), but no algorithm with single-exponential running time exists.*

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

▶ **Theorem 1.** *There is no* XP *algorithm for* Graph-Diet *with parameter d unless* P= NP.

3.2 Lower Bounds for Tree-Diet

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

▶ **Theorem 2.** Tree-Diet *and* Path-Diet *are* Para-NP*-hard for the target treewidth parameter tw' (i.e.* NP*-hard for $tw' = 1$).*

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

▶ **Theorem 3.** Tree-Diet *is* Para-NP*-hard for parameter d. More precisely, it is* W[1]*-hard for parameter Δ 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 δ 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

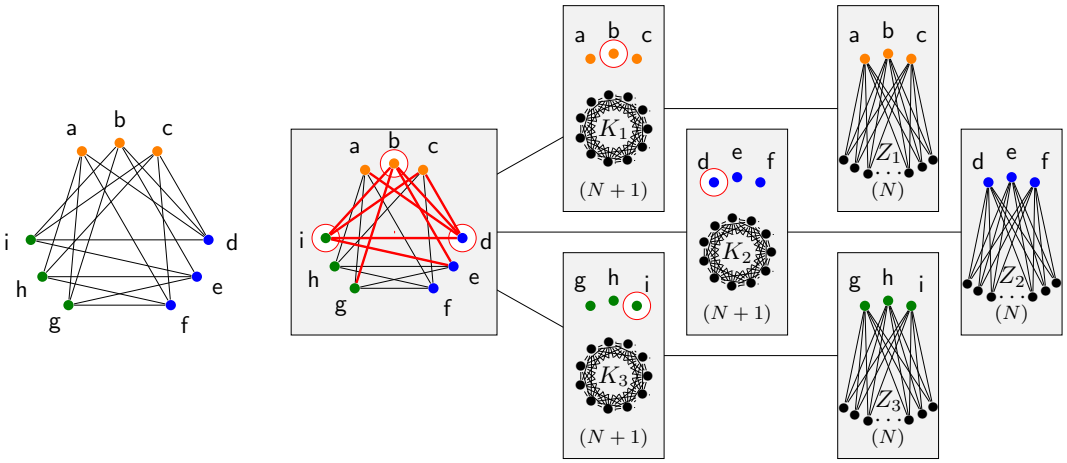


Figure 4 Reduction for Theorem 3 showing that Tree-Diet is NP-hard even for $d = 1$, from a graph G (left) with $k = 3$ and $n = 3$ to a graph G' (right, given by its tree-decomposition of width $N + n + 1$): a 1-tree diet for G' amounts to selecting a k-clique in the root bag, i.e. in G.

$\mathcal{T}'$: start with $G' := G$. Add k independent cliques $K_1, \ldots, K_k$ 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.

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

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

▶ **Theorem 4.** Path-Diet *is* W[1]*-hard for parameter* d.

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

4 FPT Algorithm

4.1 For general tree-decompositions

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

4.1.1 Coloring formulation

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

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

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

These rules are summarized in Figure 5 (a).

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

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

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

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

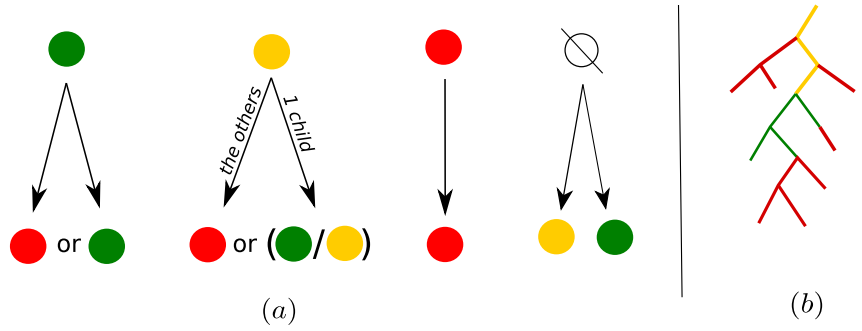


Figure 5 (a) Color assignation rules for vertices, when going down-tree. (b) Sketch of the general pattern our color assignation rules create on $\mathcal{T}_u$, the sub-tree of bags containing a given vertex u

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

4.1.2 Decomposition of the search space and sub-problems

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

Consider an edge (u, v) of G, realizable when coloring a tree-decomposition $\mathcal{T}$ of G with $\mathcal{C}$, we write $\mathcal{T}_{uv}^{\mathrm{g}}$ the subtree of $\mathcal{T}$ in which both u and v are green. We denote by $\mathcal{T}_i$ the sub-tree-decomposition rooted at X_i, G_i the subgraph induced by the vertices present in $\mathcal{T}_i$, and $C(i, f)$ the colorings of $\mathcal{T}_i$ agreeing with f on i. Our dynamic programming table is then defined as:

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

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 $\mathcal{T}_{uv}^{\mathrm{g}}$ is entirely contained strictly below X_i. We have:

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

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

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

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

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

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

with

- *m: a map from the orange vertices in X_i to the children of X_i. It decides for each orange vertex u, which child, among those which contain u, will color u orange or green;*
- *$comp(Y_j, f, m)$: the set of colorings of Y_j compatible with f on X_i and m;*
- *$count(f, f'_j)$: set of edges of G involving two vertices of Y_j green by f'_j, but such that one of them is either not in X_i or not green by f.*

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

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

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

4.1.2.1 Dynamic programming algorithm

The recurrence relation of Theorem 6 naturally yields a dynamic programming algorithm for the Tree-Diet problem, as stated below:

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

The proof of this theorem is postponed to Appendix C.

▶ **Corollary 9.** Binary-Tree-Diet *($\Delta = 2$) admits an* FPT *algorithm for the tw parameter.*

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

4.2 For path decompositions

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

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

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

▶ **Theorem 11.** Path-Diet *can be solved in $O(tw^{2d}n)$-space and $O(tw^{4d}n)$-time.*

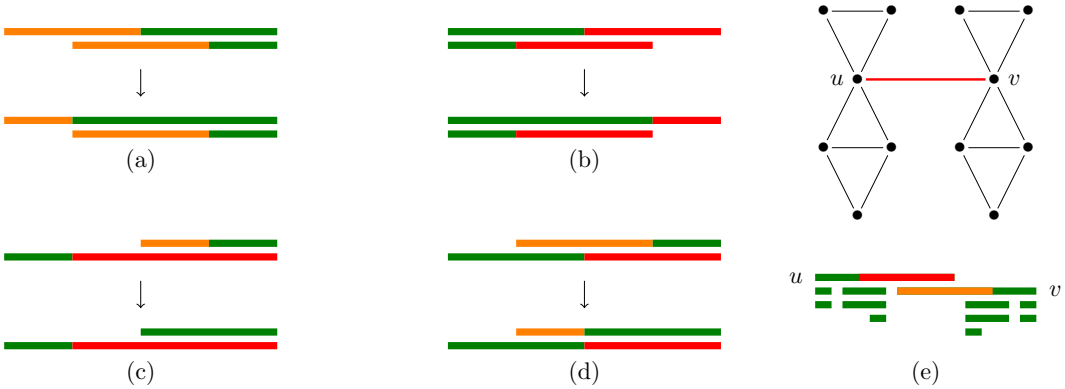


Figure 6 Five cases where two vertices are deleted in the same bag with $d = 1$. Bags are points in the line, and an interval covering all bags containing v is drawn for each v (with an equivalent coloring, see Proposition 5). Cases (a) to (d) can be safely avoided by applying the given transformations.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

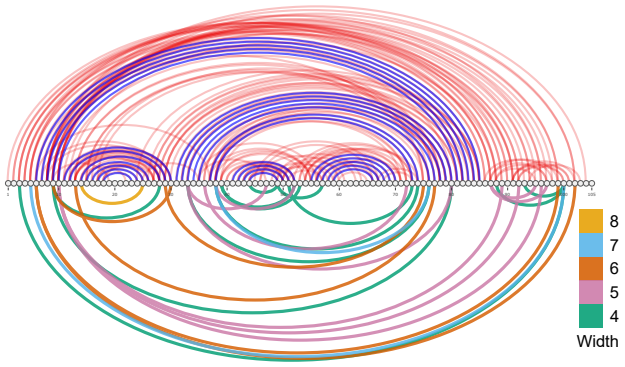
5 Proofs of concept

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

5.1 Memory-parsimonious unbiased sampling of RNA designs

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

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



tw'	#Preserved BPs EteRNA22	EteRNA77
9	–	**183**
8	–	182
7	–	180
6	**465**	176
5	460	168
4	456	157
3	445	144
2	418	121
1	320	86

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.

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

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

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

5.2 Structural alignment of complex RNAs

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

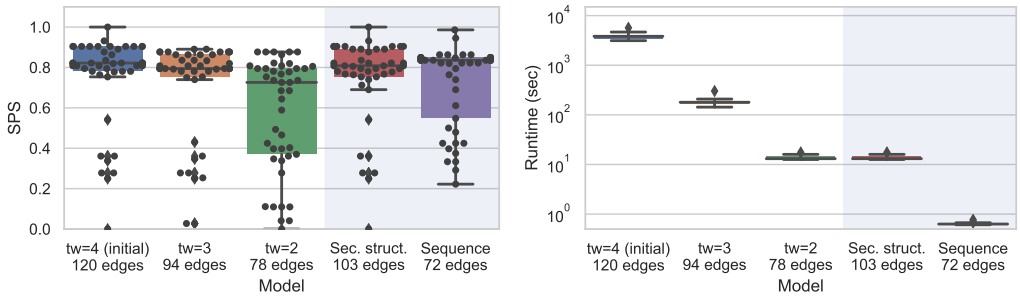


Figure 8 Impact on alignment quality (SPS; Left) and runtime (Right) of simplified instances for the RNA sequence-structure alignment of the pseudoknotted c-di-GMP-II riboswitch. The impact of simplifications on the quality of predicted alignments, using RFAM RF01786 as a reference, appears limited while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

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

5.2.1 Impact of treewidth on the structural alignment of a riboswitch

In this case study, we used our tree-diet algorithm to modulate the treewidth of complex RNA structures, and investigate the effect of the simplification on the quality and runtimes of structure-sequence alignments. We considered the Cyclic di-GMP-II riboswitch, a regulatory motif found in bacteria that is involved in signal transduction, and undergoes conformational change upon binding the second messenger c-di-GMP-II [32, 33]. A 2.5Å resolution 3D model of the c-di-GMP-II riboswitch in *C. acetobutylicum*, proposed by Smith *et al* [30] based on X-ray crystallography, was retrieved from the PDB [4] (PDBID: 3Q3Z). We annotated its base pairs geometrically using the DSSR method [22]. The canonical base pairs, supplemented with the backbone connections, were then accumulated in a graph, for which we heuristically computed an initial tree decomposition $\mathcal{T}_4$, having treewidth $tw = 4$.

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

$$\mathsf{SPS}(a_{\mathcal{T},w}; a^\star_w) = \frac{|\,\mathsf{MatchedCols}(a_{\mathcal{T},w}) \cap \mathsf{MatchedCols}(a^\star_w)\,|}{|\,\mathsf{MatchedCols}(a^\star_w)\,|},$$

using as reference the alignment $a^\star_w$ between the 3Q3Z sequence and w induced by the manually-curated RFAM alignment of the RF01786 family.

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

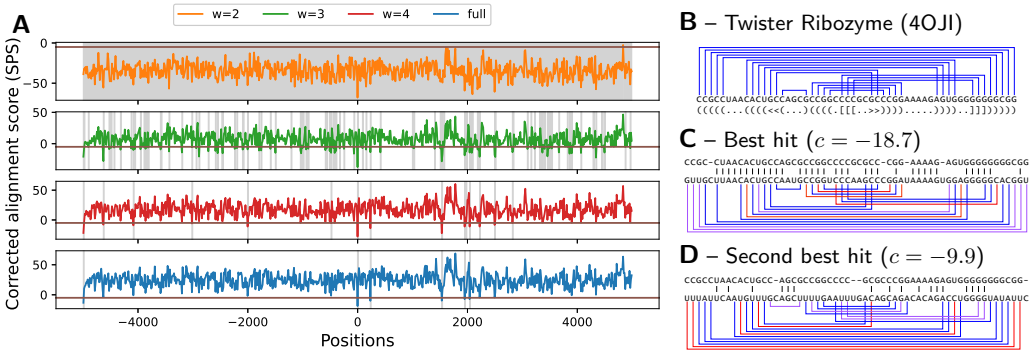


Figure 9 Corrected costs associated with the search for structured homologs of the Twister ribozyme in chromosome 5 of *S. bicolor*, using simplified instances of various treewidth (A). Gray areas represent scores which, upon correction, remain below the cutoff, and have to be considered for further 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).

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

5.2.2 Exact iterative strategy for the genomic search of ncRNAs

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

Our instance simplification framework enables an alternative strategy, that incrementally filters out unsuitable windows based on models of increasing granularity. Indeed, for any given target sequence, the min alignment cost c_δ obtained for a simplified instance of treewidth $tw - \delta$ can be corrected (*cf* Appendix G) into a lower bound $c^\star_\delta$ for the min alignment cost $c^\star_0$ of the full-treewidth instance tw. Any window such that $c^\star_\delta > \varepsilon$ thus also obeys $c^\star_0 > \varepsilon$, and can be safely discarded from the list of putative ε-admissible windows, without having to perform a full-treewidth alignment. Given the exponential growth of the alignment runtime for increasing treewidth values (see Figure 8-right) this strategy is expected to yield substantial runtime savings.

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

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

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

6 Conclusion and discussion

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

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

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

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

References

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

19 Robert J Klein and Sean R Eddy. Rsearch: finding homologs of single structured RNA sequences. *BMC bioinformatics*, 4(1):44, 2003.

20 Neocles B Leontis and Eric Westhof. Geometric nomenclature and classification of RNA base pairs. *RNA*, 7(4):499–512, 2001.

21 Yijin Liu, Timothy J Wilson, Scott A McPhee, and David MJ Lilley. Crystal structure and mechanistic investigation of the twister ribozyme. *Nature chemical biology*, 10(9):739–744, 2014.

22 Xiang-Jun Lu, Harmen J. Bussemaker, and Wilma K. Olson. DSSR: an integrated software tool for dissecting the spatial structure of RNA. *Nucleic Acids Research*, 43(21):e142–e142, 07 2015. arXiv:https://academic.oup.com/nar/article-pdf/43/21/e142/17435026/gkv716.pdf, doi:10.1093/nar/gkv716.

23 R. B. Lyngsø and C. N. S. Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology*, 7(3-4):409–427, 2000.

24 Vladimir Reinharz, Antoine Soulé, Eric Westhof, Jérôme Waldispühl, and Alain Denise. Mining for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network families. *Nucleic Acids Research*, 46(8):3841–3851, mar 2018. doi:10.1093/nar/gky197.

25 Philippe Rinaudo, Yann Ponty, Dominique Barth, and Alain Denise. Tree decomposition and parameterized algorithms for RNA structure-sequence alignment including tertiary interactions and pseudoknots. In *Lecture Notes in Computer Science*, pages 149–164. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-33122-0_12.

26 Elena Rivas and Sean R Eddy. Parameterizing sequence alignment with an explicit evolutionary model. *BMC bioinformatics*, 16(1):406, 2015.

27 Toshiki Saitoh, Ryo Yoshinaka, and Hans L. Bodlaender. Fixed-treewidth-efficient algorithms for edge-deletion to interval graph classes. In *WALCOM: Algorithms and Computation - 15th International Conference and Workshops, 2021, Proceedings*, volume 12635 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2021. doi:10.1007/978-3-030-68211-8_12.

28 Roman Sarrazin-Gendron, Hua-Ting Yao, Vladimir Reinharz, Carlos G. Oliver, Yann Ponty, and Jérôme Waldispühl. Stochastic sampling of structural contexts improves the scalability and accuracy of RNA 3d module identification. In *Lecture Notes in Computer Science*, pages 186–201. Springer International Publishing, 2020. doi:10.1007/978-3-030-45257-5_12.

29 Saad Sheikh, Rolf Backofen, and Yann Ponty. Impact Of The Energy Model On The Complexity Of RNA Folding With Pseudoknots. In Juha Kärkkäinen and Jens Stoye, editors, *CPM - 23rd Annual Symposium on Combinatorial Pattern Matching - 2012*, volume 7354 of *Combinatorial Pattern Matching*, pages 321–333, Helsinki, Finland, July 2012. Juha Kärkkäinen, Springer. doi:10.1007/978-3-642-31265-6_26.

30 Kathryn D. Smith, Carly A. Shanahan, Emily L. Moore, Aline C. Simon, and Scott A. Strobel. Structural basis of differential ligand recognition by two classes of bis-(3'-5')-cyclic dimeric guanosine monophosphate-binding riboswitches. *Proceedings of the National Academy of Sciences*, 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, doi:10.1073/pnas.1018857108.

31 Yinglei Song, Chunmei Liu, Russell Malmberg, Fangfang Pan, and Liming Cai. Tree decomposition based fast search of RNA structures including pseudoknots in genomes. In *Computational Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE*, pages 223–234. IEEE, 2005.

32 N. Sudarsan, E. R. Lee, Z. Weinberg, R. H. Moy, J. N. Kim, K. H. Link, and R. R. Breaker. Riboswitches in eubacteria sense the second messenger cyclic di-gmp. *Science*, 321(5887):411–413, 2008. URL: https://science.sciencemag.org/content/321/5887/411, arXiv:https://science.sciencemag.org/content/321/5887/411.full.pdf, doi:10.1126/science.1159519.

33 Rita Tamayo. Cyclic diguanylate riboswitches control bacterial pathogenesis mechanisms. *PLOS Pathogens*, 15(2):1–7, 02 2019. doi: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/
587 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.
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.

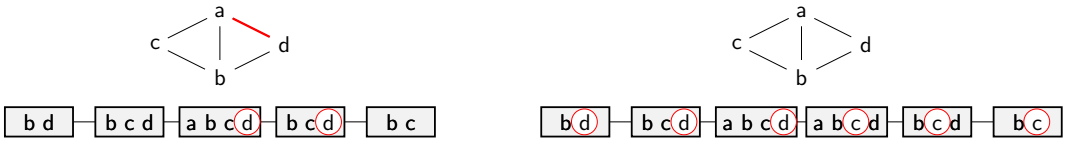


Figure 10 Left: A graph and a path-decomposition whose optimal 1-tree diet loses an edge (ad). However, duplicating the bag abcd (right) yields a tree-decomposition with a lossless 1-tree diet.

A Editing Trees before the Diet

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

B Postponed Proofs in Section 3.2

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

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

C Postponed proofs of dynamic programming scheme

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

$$count(f,h) = \left\{ (u,v) \in E \;\middle|\; \begin{array}{c} h(u) = h(v) = green \quad \text{and} \\ (u \notin f^{-1}(*) \text{ or } f(u) \neq g \text{ or } v \notin f^{-1}(*) \text{ or } f(v) \neq g) \end{array} \right\}$$

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

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

Proof of Theorem 6.

$\boxed{\leq}$ By definition, $c(X_i, f) = \max_{\mathcal{C} \in C(\mathcal{T}_i, f)} |RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)|$ is the maximum number of realizable edges in the sub-tree-decomposition rooted at X_i, such that all green-green occurences of the edge occur strictly below X_i, and under the constraint that f colors X_i. Let $\mathcal{C}$ be a coloring for $\mathcal{T}_i$ realizing the optimum $c(X_i, f)$. Its restrictions to $Y_1 \dots Y_\Delta$ yield colorings $f'_1 \dots f'_\Delta$. Likewise, its restrictions to the sub-tree-decompositions $\mathcal{T}'_1 \dots \mathcal{T}'_\Delta$ rooted at $Y_1 \dots Y_\Delta$ yield colorings $\mathcal{C}'_1 \dots \mathcal{C}'_\Delta$ compatible with $f'_1 \dots f'_\Delta$. $\mathcal{C}'_1 \dots \mathcal{C}'_\Delta$ cannot be better than the optimal, so $\forall j$, $|RE_{\downarrow}(\mathcal{T}'_j, \mathcal{C}'_j, G)| \leq c(Y_j, f'_j)$
Let (u,v) be an edge of $RE_{\downarrow}(\mathcal{T}_i, \mathcal{C}, G)$. Per Lemma 7, either $(u,v) \in count(f, f'_j)$ for some j (if Y_j is the root of $\mathcal{T}^{\mathrm{g}}_{uv}$) and $(u,v) \notin \cup_j RE_{\downarrow}(\mathcal{T}'_j, \mathcal{C}'_j, G)$ or $(u,v) \in count(f, f'_j)$ and $\exists j$ such that $(u,v) \in RE_{\downarrow}(\mathcal{T}'_j, \mathcal{C}'_j, G)$. Therefore:

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

and, a fortiori

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

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

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

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

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

which concludes the proof.

◀

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

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

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

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

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

D Postponed proofs of path decomposition case

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

E Pseudo-code

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

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

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

Edge weights are passed to the function in the form of a dictionary/map W associating a real weight to each edge. Within Algorithms 1 and 2, the only place where it is taken into account is the the *count* function, which computes the weight of edges accounted for by the 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) ∪ f^{-1}(r)| ≤ (|X_i| − tw' − 1) then
4          //not enough removals.;
5          c[X_i, f] = −∞;
6          return c[X_i, f];
7      end
8      if X_i == leaf then
9          c[X_i, f] = 0;
10         return c[X_i, f];
11     end
12     int ans = −∞;
13     for m ∈ orange_maps(X_i, f) do
14         int ans_m = 0;
15         for Y_j ∈ X_i.children do
16             int ans_j = −∞;
17             for f'_j ∈ compatible(f, m, X_i, Y_j) do
18                 int val = 0;
19                 val += count(f, f'_j, W);
20                 val += optim_num_real_edges(Y_j, f'_j, G, tw');
21                 if val ≥ ans_j then
22                     ans_j = val;
23                 end
24             end
25             ans_m += ans_j;
26         end
27         if ans_m ≥ ans then
28             ans = ans_m;
29         end
30     end
31     c[X_i, f] = 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 coloration for $\mathcal{T}$

```
1  Function optim_coloring(X_i, f, G, tw', c):
2      if X_i == leaf then
3          return ∅;
4      end
5      coloring C = ∅;
6      for m ∈ orange_maps(X_i, f) do
7          int ans_m = 0;
8          coloring best_fjs = [];
9          for Y_j ∈ X_i.children do
10             int best_valj = −∞;
11             int best_fj = ∅;
12             for f'_j ∈ 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 ≥ 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 C;
31 end
```

F Correctness of the rejection-based sampling of candidate RNA designs

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

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

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

where $\beta > 0$ is an arbitrary constant akin to a temperature. Yao *et al* describe an algorithm which generates k sequences in $\Theta(k(n + |\mathcal{D}|))$ time, after a preprocessing in $\Theta(n.|\mathcal{D}|.4^{tw})$ time and $\Theta(n.4^{tw})$ 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 $p := e^{-\beta.E_{w,S}}/\mathcal{Z}_{S',\mathcal{D}'}$ 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

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

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

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

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

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

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

Let S be an arc-annotated sequence of length m (S_i denotes the ith character of S), w be a target (flat) sequence of length m, and $\mu : [1, n] \to [1, m] \cup \{\perp\}$ represents an alignment[2]. We consider the following cost function, adapted from [25], which quantifies the quality of an alignment μ for S and w:

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

where

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

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

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

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

▶ **Proposition 12.**

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

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

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

Minimizing over all alignment μ, one obtains

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

[2] An alignment μ 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 is this discussion for the sake of simplicity.

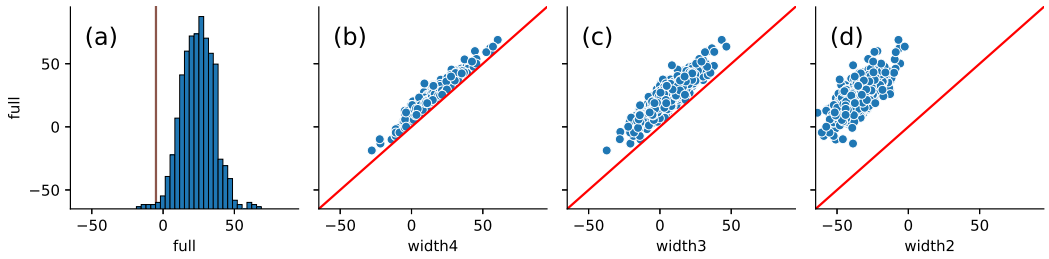


Figure 11 (a) Histogram of alignment scores obtained by aligning the full structure ($tw = 5$) model of the Twister ribozyme (pdb-id: 4OJI) with $\kappa \cdot n$-sized windows in a 10kb region of the 5^{th} chromosome of *S. bicolor*. A vertical line is positioned at the ϵ threshold. (b;c;d) Corrected 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.

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

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

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

$$\begin{aligned} c^\star &\geq c' - \sum_{\substack{i \text{ unpaired in } S,' \\ \text{paired in } S}} \max_{\mu} \gamma(S_i, w_k) + \sum_{(i,j)\in S\setminus S'} \min_{\mu} \phi(S_i, S_j, w_k, w_l) \\ &= c' - \sum_{\substack{i \text{ unpaired in } S,' \\ \text{paired in } S}} \max_{a} \gamma(S_i, a) + \sum_{(i,j)\in S\setminus S'} \min_{a,b} \phi(S_i, S_j, a, b). \end{aligned}$$

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

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