RESEARCH

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

Bertrand Marchand[1,2], Yann Ponty[1*] and Laurent Bulteau[2*]

*Correspondence:
yann.ponty@lix.polytechnique.fr;
laurent.bulteau@u-pem.fr
[1]LIX CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
[2]LIGM, CNRS, Univ Gustave Eiffel, F77454 Marne-la-Vallée, France
Full list of author information is available at the end of the article

Abstract

Hard graph problems are ubiquitous in Bioinformatics, inspiring the design of specialized Fixed-Parameter Tractable algorithms, many of which rely on a combination of tree-decomposition and dynamic programming. The time/space complexities of such approaches hinge critically on low values for the treewidth tw of the input graph. In order to extend their scope of applicability, we introduce the Tree-Diet problem, *i.e.* the removal of a minimal set of edges such that a given tree-decomposition can be slimmed down to a prescribed treewidth tw'. Our rationale is that the time gained thanks to a smaller treewidth in a parameterized algorithm compensates the extra post-processing needed to take deleted edges into account.

Our core result is an FPT dynamic programming algorithm for Tree-Diet, using $2^{O(tw)}n$ time and space. We complement this result with parameterized complexity lower-bounds for stronger variants (e.g., NP-hardness when tw' or $tw - tw'$ is constant). We propose a prototype implementation for our approach which we apply on difficult instances of selected RNA-based problems: RNA design, sequence-structure alignment, and search of pseudoknotted RNAs in genomes, revealing very encouraging results. This work paves the way for a wider adoption of tree-decomposition-based algorithms in Bioinformatics.

Keywords: RNA; treewidth; FPT algorithms; RNA design; structure sequence alignment

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 [1], structural bioinformatics [2], comparative genomics [3], phylogenetics [4], and further examples that can be found in a review by Bulteau and Weller [5]. RNA bioinformatics is no exception, with the prevalence of the secondary structure, an outer planar graph [6], as an abstraction of RNA conformations, and the notable utilization of graph models to represent complex topological motifs called pseudoknots [7], inducing the hardness of several tasks, such as structure prediction [8, 9, 10], structure alignment [11], or structure/sequence alignment [12]. Such motifs are functionally important and conserved, as witnessed by their presence in the consensus structure of 336 RNA families in the 14.5 edition of the RFAM database [13]. Moreover, methods in RNA bioinformatics [14] are increasingly considering non-canonical base pairs and modules [15, 16], further increasing the density of RNA structural graphs and outlining the need for scalable algorithms.

A parameterized complexity approach can be used to circumvent the frequent NP-hardness of relevant problems. It generally considers one or several parameters, whose values are naturally bounded (or much smaller than the input size) within real-life instances. Once relevant parameters have been identi ed, one aims to design a Fixed Parameter Tractable (FPT) algorithm, having polynomial complexity for any xed 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 e cient heuristics [17, 18] for computing tree-decompositions of reasonable treewidth. Given a tree-decomposition, many combinatorial optimization tasks can be solved using dynamic programming (DP), in time/space complexities that remain polynomial for any xed treewidth value. Resulting algorithms remain correct upon (almost) arbitrary modi cations 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-e cient implementation, even when the parameter takes low typical values. Indeed, the dependency of the complexity on the treewidth may be prohibitive, both in terms of time and memory requirements. This limitation is particularly obvious while searching and aligning structured RNAs, giving rise to an algorithmic problem called RNA structure-sequence alignment [19, 20, 12], for which the best known exact algorithm is in $(n{:}m^{tw+1})$, with n the structure length, m the sequence/window length, and tw the treewidth of the structure (inc. backbone). Such a complexity becomes impractical for structures having a treewidth higher than 4, which represent 50 to 70% of known RNA structures, as shown by Figure 1, based on a broad analysis of structures found in the PDB database. Similar complexities hold for problems that can be expressed as (weighted) constraint satisfaction problems, with m representing the cardinality of the variable domains. Such frameworks are frequently used for molecular design, both in proteins [21] and RNA [22], and may require the consideration of tree-widths as high as 20 or more [23].

In this paper, we investigate a pragmatic strategy to increase the practicality of parameterized algorithms based on the treewidth parameter [27]. We put our instance graphs on a diet, i.e. we introduce a preprocessing that reduces their treewidth to a prescribed value by removing a minimal cardinality set of edges. As discussed previously, the practical complexity of many algorithms greatly benets from the consideration of simpli ed instances, having lower treewidth. Moreover, speci c countermeasures for errors introduced by the simpli cation can sometimes be used to preserve the correctness of the algorithm. For instance, for searching structured RNAs using RNA structure-sequence alignment [19], an iterated ltering 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 simpli ed instances in RNA design. An overview of our approach is sketched on Figure 2

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

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, a practical Dynamic Programing FPT algorithm for Tree-Diet , along with possible further optimizations for Path-Diet , two natural simpli cations 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 T (over a set V of vertices) is a tree whose nodes are subsets of V, known as bags. The bags containing any $v \in V$ induce a (connected) subtree of T. A path-decomposition is a tree-decomposition whose underlying tree T is a path. The width of T (denoted $w(T)$) is the size of its largest bag minus 1. An edge $\{u, v\}$ is visible in T if some bag contains both u and v, otherwise it is lost. T is a tree-decomposition of G if all edges of G are visible in 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 $tw' < tw$, nd a tree-decomposition over V of width at most tw' losing a minimum number of edges from G.

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

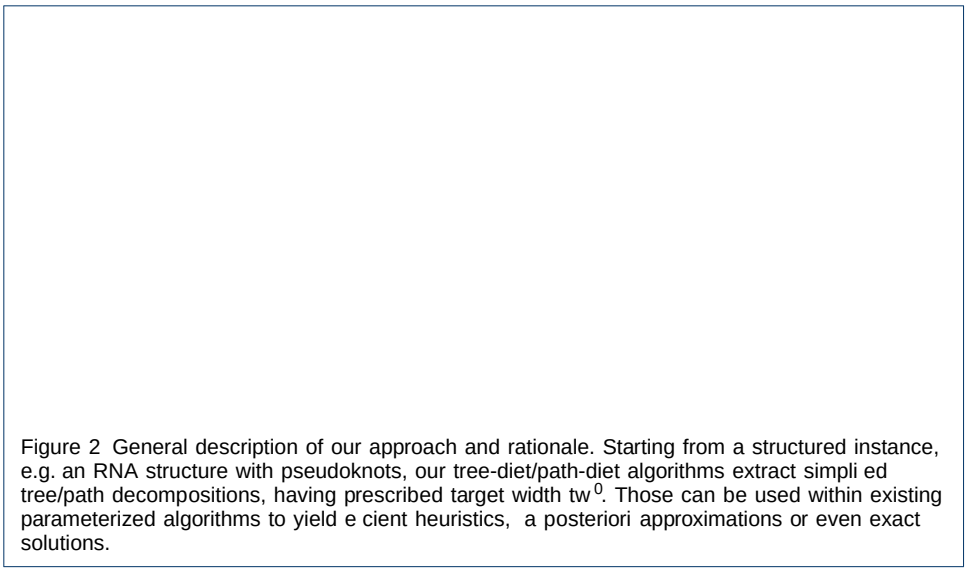

Figure 2 General description of our approach and rationale. Starting from a structured instance, e.g. an RNA structure with pseudoknots, our tree-diet/path-diet algorithms extract simpli ed tree/path decompositions, having prescribed target width tw^0. Those can be used within existing parameterized algorithms to yield e cient heuristics, a posteriori approximations or even exact solutions.

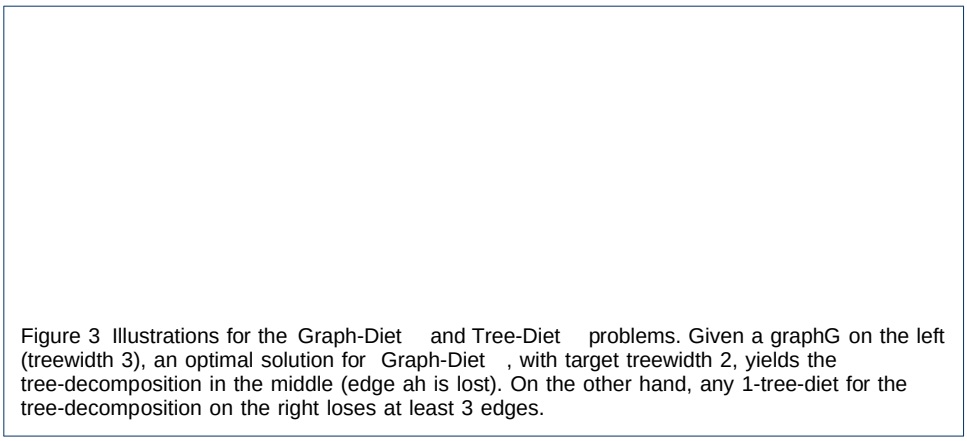

Figure 3 Illustrations for the Graph-Diet and Tree-Diet problems. Given a graph G on the left (treewidth 3), an optimal solution for Graph-Diet, with target treewidth 2, yields the tree-decomposition in the middle (edge ah is lost). On the other hand, any 1-tree-diet for the tree-decomposition on the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

Problem (Tree-Diet) Given a graph G, a tree-decomposition T of G of width tw, and an integer $tw^0 < tw$, nd a $(tw \quad tw^0)$-tree-diet of T losing a minimum number of edges.

Note that for Tree-Diet, T does not have to be optimal, so the width tw of the input tree decomposition might be larger than the actual treewidth of G, thus Tree-Diet can be used to reduce the width of any input decomposition. We de ne Binary-Tree-Diet and Path-Diet analogously, where T is restricted to be a binary tree (respectively, a path). An example of an instance of Graph-Diet and of Tree-Diet are given in Figure 3.

Parameterized Complexity in a Nutshell The basics of parameterized complexity can be loosely de ned as follows (see [28] for the formal background). A parameter k for a problem is an integer associated with each instance which is expected to remain small in practical instances (especially when compared to the input size n). An exact algorithm, or the problem it solves, is FPT if it takes time $f(k)\text{poly}(n)$, and XP if it takes time $n^{g(k)}$ (for some functions $f; g$). Under commonly accepted conjectures (see for instance [29] for details) W[1]-hard problems may not be FPT,

<table>
<tr><th>Parameter
Problem</th><th colspan="2">Source treewidth
tw</th><th>Target treewidth
tw^0</th><th>Difference
$d = tw - tw^0$</th></tr>
<tr><td>Graph-Diet</td><td colspan="2">FPT
via MSO
Theorem 1</td><td>Para-NP-hard
$tw^0 = 2$
EDP(K_4) [30]</td><td>Para-NP-hard*
$d = 1$
Theorem 2</td></tr>
<tr><td>Tree-Diet</td><td>XP
$O((6\Delta)^{tw})$
Theorem 7</td><td>FPT open</td><td rowspan="3">Para-NP-hard
$tw^0 = 1$
Theorem 3</td><td>Para-NP-hard
$d = 1$
Theorem 4</td></tr>
<tr><td>Binary-Tree-Diet</td><td colspan="2" rowspan="2">FPT
$O(12^{tw})$
Theorem 7</td><td rowspan="2">W[1]-hard
Theorem 5</td><td>XP open</td></tr>
<tr><td>Path-Diet</td><td>XP
$O(tw^d)$
Theorem 8</td></tr>
</table>

Table 1 Parameterized results for our problems. Algorithm complexities are given up to polynomial time factors (O notation), Δ denotes the maximum number of children in the input tree-decomposition. (*) see Theorem 2 statement for a more precise formulation.

and Para-NP-hard problems (NP-hard even for some fixed value of k) are not 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).

Thus we focus on the Tree-Diet relaxation, where an input tree-decomposition is given, which we use as a guide/restriction towards a thinner tree-decomposition. Seen as an additional constraint, it makes the problem harder (the case $tw^0 = 1$ becomes NP-hard, Theorem 3, although for Graph-Diet it corresponds to the Spanning Tree problem and is polynomial). With parameter tw however, it does help reduce the search space. In Theorem 7 we give an $O((6\Delta)^{tw^2} n)$ 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 Supplementary Section A for more details). Moreover, the duplications of bags which are used in the conversion may only decease the number of lost edges incurred by Tree-Diet.

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

3 Algorithmic Limits: Parameterized Complexity Considerations

Graph-Diet can be seen as a special case of the Edge Deletion Problem (EDP) for the family of graphs H of treewidth tw^0 or less: given a graph G, remove as few edges as possible to obtain a graph in H. Such edge modification problems are more often parameterized by the number k of edited edges (see [31] for a complete survey). Given our focus on increasing the practicality of treewdith-based algorithms in bioinformatics, we restrict our focus to treewidth related parameters tw, tw^0 and $d = tw - tw^0$.

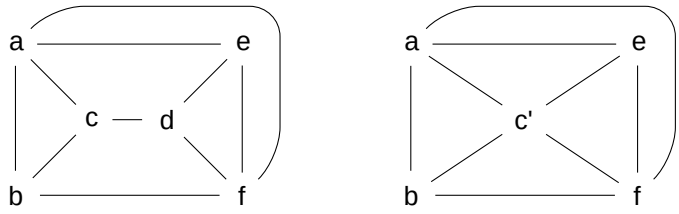

Figure 4 A graph G (left) with treewidth 3. Deleting edge cd gives treewidth 2, implying that G 2 Treewidth2 + 1e. However, if one contracts edge cd, then the resulting graph (right) has treewidth 3, and deleting any single edge does not decrease the treewidth. This example shows that the graph family Treewidth 2+1e is not minor-closed.

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

3.1 Graph-Diet: practical solutions seem unlikely

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

Regarding the source graph treewidth tw, the vertex deletion equivalent of Graph-Diet, where one asks for a minimum subset of vertices to remove to obtain a given treewidth, is known as a Treewidth Modulator. This problem has been better-studied than its edge-deletion counterpart [35], and has been shown to be FPT for the treewidth [36]. For the edge-deletion version (Graph-Diet), we can use an optimization variant of Courcelle's Theorem [29, Thm. 7.12] to show that the problem is FPT for tw. However, this is a purely theoretical result as the running-time of such "black-box" algorithms typically involve towers of exponentials on the treewidth parameter.

Theorem 1 Graph Diet is FPT for the treewidth.

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

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

Conjecture 1 Graph-Diet does not admit algorithms with single-exponential running time for the treewidth.

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

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

Theorem 2 There is no XP algorithm for Graph-Diet with parameter d unless P= NP.

Proof We consider the decision version of Graph-Diet where a bound k on the number of deleted edges is given. We build a Turing reduction from Treewidth : more precisely, assuming an oracle for Graph-Diet with $d = 1$ is available, we build a polynomial-time algorithm to compute the treewidth of a graph G. This is achieved by computing Graph-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). □

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 T has a constant degree remains open (cf. Table 1).

Theorem 3 Tree-Diet and Path-Diet are Para-NP-hard for the target treewidth parameter tw' (NP-hard for $tw' = 1$).

Proof By reduction from the NP-hard problem Spanning Caterpillar Tree [40]: given a graph G, does G have a spanning tree C that is a caterpillar? Given $G = (V; E)$ with $n = |V|$, we build a tree-decomposition T of G consisting of $n - 1$ bags containing all vertices (the width of the decomposition is therefore $n - 1$) connected in a path. Then $(G; T)$ admits a tree-diet to treewidth 1 with $n - 1$ visible

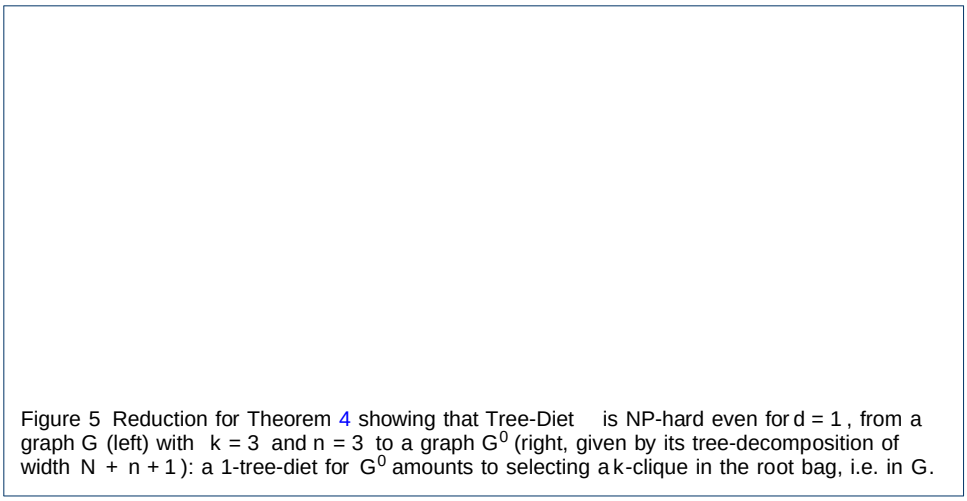

Figure 5 Reduction for Theorem 4 showing that Tree-Diet is NP-hard even for $d = 1$, from a graph G (left) with $k = 3$ and $n = 3$ to a graph G^0 (right, given by its tree-decomposition of width $N + n + 1$): a 1-tree-diet for G^0 amounts to selecting a k-clique in the root bag, i.e. in G.

edges if, and only if, G admits a caterpillar spanning tree. Indeed, the subgraph of G with visible edges must be a graph with pathwidth 1, i.e. a caterpillar [41]. With $n - 1$ visible edges, the caterpillar connects all n vertices together, i.e. it is a spanning tree. □

Theorem 4 Tree-Diet is Para-NP-hard for parameter d. More precisely, it is W[1]-hard for parameter Δ, the degree of T, even when $d = 1$.

Proof By reduction from Multi-Colored Clique (Given a graph G, an integer k and a partition of the vertices of G into k sets, is there a clique in G containing exactly one vertex from each of the k sets?). Consider a k-partite graph $G = (V; E)$ with $V = \bigcup_{i=1}^{k} V_i$. We assume that G is regular (each vertex has degree and that each V_i has the same size n (Multi Colored Clique is W[1]-hard under these restrictions [28, 29]). Let $L := k \binom{k}{2}$ and $N = \max\{|V_i|; L + 1\}$. We now build a graph G^0 and a tree-decomposition T^0: start with $G^0 := G$. 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 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, T is a tree-decomposition of G^0 with $\Delta = k$ and maximum bag size $n + N + 1$ (vertices of V induce a size-3 path in 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:

T has a 1-tree-diet losing at most L edges from G^0, $\Leftrightarrow$ G has a k-clique.

($\Rightarrow$) Assume G has a k-clique $X = \{x_1; \ldots; x_k\}$ (with $x_i \in V_i$). Build T^0 by removing each x_i from bags T_0 and $T_{1;i}$. Then T^0 is a 1-tree-diet of T. There are no edges lost by removing x_i from $T_{1;i}$ (since x_i is not connected to K_i), and the edges lost in T_0 are all edges of G adjacent to any x_i. Since X forms a clique and each x_i has degree Δ, there are $L = k \binom{k}{2}$ such edges.

) Consider a 1-tree-diet T' of T losing L edges. Since each bag $T_{1;i}$ has maximum size, 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 $N - L + 1$ edges). Furthermore, x_i may not be removed from $T_{2;i}$ (otherwise N edges between x_i and Z_i would be lost), so x_i must also be removed from T_0. Let K be the number of edges in $G[\{x_1, \ldots, x_k\}]$. The total number of lost edges in T_0 is $k - K$. Thus, we have $k - K \leq L$ and $K \geq \binom{k}{2}$: $\{x_1, \ldots, x_k\}$ form a k-clique of G. □

Theorem 5 Path-Diet is W[1]-hard for parameter d.

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

4 FPT Algorithm

4.1 For general tree-decompositions

We describe here a $O(3^{tw} n)$-space, $O(^{tw+2} 6^{tw} n)$-time dynamic programming algorithm for the Tree-Diet problem, with 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 T of width tw of a graph G, we want to remove vertices from the bags of T to reach a target width tw' while losing as few edges from G as possible. We tackle the problem through an equivalent coloring formulation: our algorithm will assign a color to each occurrence of a vertex in the tree decomposition. We work with three colors: red (r), orange (o), and green (g). Green means that the vertex is kept in the bag, while orange and red means removal of the vertex. An edge is thus visible within a bag when both its ends are green. It is lost if there is no bag where it is visible. To ensure equivalence with the original problem, the colors will be assigned following local rules, which we now describe.

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

- A vertex of a bag not present in its parent may be green or orange (R1)
- A green vertex in a bag may be either green or red in its children (R2)
- A red vertex in a bag must stay red in its children (R3)

- An orange vertex in a bag has to be either orange or green in exactly one child (unless there is no child with this vertex), and must be red in the other children (R4)

These rules are summarized in Figure 6 (a).

When going down the tree, a green vertex may only stay green or permanently become red. As for orange vertices, they are locally absent but \may potentially be found further down the tree", while red vertices are removed from both the current bag and its entire subtree. An immediate consequence of these rules is therefore that the green occurences of a given vertex form a (possibly empty) connected subtree. (R4) in particular is crucial to this connectivity: if an orange vertex could become orange in several children, it would be able to turn green in several disconnected subtrees. Figure 6 (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 o that path entirely red. It ends at the top of a (potentially empty) green subtree, whose vertices may also be parents to entirely red subtrees.

We will now more formally prove the equivalence of the coloring formulation to the original problem. Let us rst introduce two de nitions. Given a valid coloring 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 C. Given an integer d, a coloring C of T is said to be d diet-valid if removing red/orange vertices reduces the width of T from $w(T)$ to $w(T)$ d.

Proposition 1 Given a graph G, a tree-decomposition T of width tw, and a target width $tw^0 < tw$, The Tree-Diet problem is equivalent to nding a $(tw$ $tw^0)$-diet-valid coloring C of T allowing for a number of realizable edges in G as large as possible.

Proof Given a $(tw$ $tw^0)$-tree-diet of T specifying which vertices are removed from which bags, we rst show how to obtain a valid coloring C for T incurring the same number of lost (unrealizable) edges. Let us denote by T^0 the tree decomposition of width tw^0 obtained by applying the diet to T. To start with, a vertex u is colored green in the bags where it is not removed. By the validity of T^0 as a decomposition, this set of bags forms a connected subtree, that we denote T_u^g. We also write T_u for the subtree of bags containing u in the original decomposition T. If T_u^g and T_u do not have the same root, then u is colored orange on the the path in T from the root of T_u (included) and the root of T_u^g (excluded). Vertex u is colored red in any other bag of 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 $(tw$ $tw^0)$-tree-diet. Conversely, an equivalent $(tw$ $tw^0)$-tree-diet is obtained from a $(tw$ $tw^0)$-diet-valid coloring by removing red/orange vertices and keeping green ones. If a given vertex has no green occurences, it is entirely removed from the tree decomposition and all its edges are lost (it becomes an isolated vertex). We may add it back to the tree decomposition by introducing a new bag containing only this vertex, which we connect arbitrarily to the tree decomposition. □

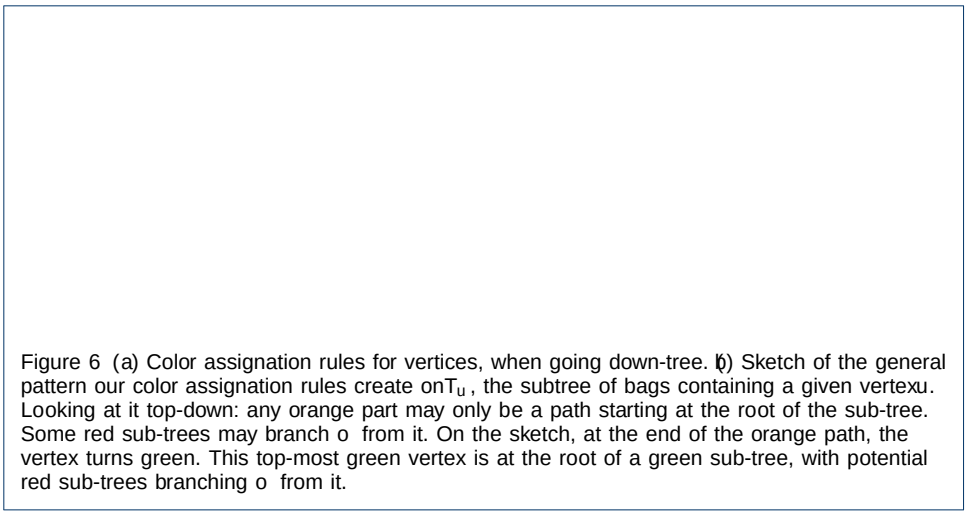

Figure 6 (a) Color assignation rules for vertices, when going down-tree. (b) Sketch of the general pattern our color assignation rules create on T_u, the subtree of bags containing a given vertex u. Looking at it top-down: any orange part may only be a path starting at the root of the sub-tree. Some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

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, $f^{-1}(g)$ denotes the green vertices of X_i, and similarly for o and r).

Let us introduce some notations before giving the definition of our dynamic programming table. Given an edge (u, v) of G, realizable when coloring a tree-decomposition T of G with C, we write T^g_{uv} the subtree of T in which both u and v are green. We denote by T_i the subtree of the decomposition rooted at X_i, and $C(i, f)$ the d-diet-valid colorings of T_i agreeing with f on i, with $d = tw - tw'$. Our dynamic programming table is then defined as:

$$c(X_i, f) = \begin{cases} \max_{C \in C(i,f)} \left\{ \begin{array}{c} \text{Edges } (u,v) \text{ of } G, \text{ realizable within } T_i \text{ colored with } C \\ \text{such that } T^g_{uv} \text{ is entirely contained strictly below } X_i \end{array} \right\} \\ \qquad \text{if } f \text{ assigns green to at most } tw' + 1 \text{ vertices} \\ 1 \qquad \text{otherwise} \end{cases}$$

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

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

Theorem 6 For a bag X_i of T, with children $Y_1, \ldots, Y_\ell$, we have:

$$c(X_i, f) = \max_{m : f^{-1}(o) \to [1..\ell]} \left[\sum_{1 \le j \le \ell} \max_{f'_j \text{ compatible } (Y_j, f, m)} c(Y_j, f'_j) + \text{count}(f, f'_j) \right]$$

with

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

Note that compatible(Y_j, f, m) may contain colorings f'_j that colour too many vertices in Y_j in green to reach target width tw'. In that case $c(Y_j, f'_j) = -1$.

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

Lemma 1 An edge (u, v) of G, realizable in T under C, is contained in exactly one set of the form count(C_{jP}, C_{jX}) with X a bag of T and P its parent, C_{jP}, C_{jX} the restrictions of C to P and X, respectively, and "count" defined as above. In addition, X is the root of the subtree of T in which both u and v are green.

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

$$\text{count}(f, h) = \{(u, v) \in E \mid h(u) = h(v) = g \text{ and } (u \notin P \text{ or } f(u) \neq g \text{ or } v \notin P \text{ or } f(v) \neq g)\}$$

Now, given a realizable edge (u, v), in a tree-decomposition T colored with C, the set of bags in which both u and v are green forms a connected subtree of T. This subtree has a root, or lowest common ancestor, that we denote $R_{(u,v)}$. Since we assumed T to be rooted at an empty bag, $R_{(u,v)}$ is not the root of 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 T, while for any other edge $X \to Y$ of T, the colors of u and v cannot verify the conditions to belong to the associated "count set". □

Proof of Theorem 6

Let us more concisely use $RE_{\#}(T_i, C, G)$ to denote the set of edges (u, v) of G, realizable under the $(tw - tw')$-diet-valid coloring C of T_i, such that T^g_{uv} is entirely contained strictly below X_i. We have, if f contains enough red/orange vertices to reduce the size of X_i to target size:

$$c(X_i, f) = \max_{C \in C(i,f)} |RE_{\#}(T_i, C, G)|$$

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

the edge occur strictly below X_i, and under the constraint that f colors X_i. Let C be a coloring for T_i realizing the optimum $c(X_i; f)$. Its restrictions to $Y_1 ::: Y$ yield colorings $f'_1 ::: f'$. Likewise, its restrictions to the subtree-decompositions $T'_1 ::: T'$ rooted at $Y_1 ::: Y$ yield colorings $C'_1 ::: C'$ compatible with $f'_1 ::: f'$. $C'_1 ::: C'$ cannot be better than the optimal, so $\forall j$, $|RE_\#(T'_j; C'_j; G)| \leq c(Y_j; f'_j)$

Let $(u; v)$ be an edge of $RE_\#(T_i; C; G)$. Per Lemma 1, either $(u; v) \in \text{count}(f; f'_j)$ for some j (if Y_j is the root of T^g_{uv}) and $(u; v) \in \bigcup_j RE_\#(T'_j; C'_j; G)$ or $(u; v) \in \text{count}(f; f'_j)$ and $\exists j$ such that $(u; v) \in RE_\#(T'_j; C'_j; G)$. Therefore:

$$c(X_i; f) = |RE_\#(T_i; C; G)| = \sum_{1 \leq j} |RE_\#(T'_j; C'_j; G)| + \text{count}(f; f'_j) \leq \sum_{1 \leq j} c(Y_j; f'_j) + \text{count}(f; f'_j)$$

and, a fortiori

$$c(X_i; f) \leq \max_{m : f^{-1}(o) \subseteq [1:::]} \sum_{1 \leq j} \max_{f'_j \in \text{compatible}(Y_j; f; m)} c(Y_j; f'_j) + \text{count}(f; f'_j)$$

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

$$\sum_{1 \leq j} c(Y_j; f'_j) + \text{count}(f; f'_j) = \sum_{1 \leq j} |RE_\#(T'_j; C'_j; G)| + \text{count}(f; f'_j) = |RE_\#(T_i; C; G)| \leq c(X_i; f)$$

This is true for any choice of $m; f'_1 ::: f'$, therefore:

$$\max_{m : f^{-1}(o) \subseteq [1:::]} \sum_{1 \leq j} \max_{f'_j \in \text{compatible}(Y_j; f; m)} c(Y_j; f'_j) + \text{count}(f; f'_j) \leq c(X_i; f)$$

which concludes the proof.

□

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

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

Proof of Theorem 7 Given the coloring formulation and Proposition 1, and given the sub-problems and $c(X_i; f)$-table de nitions, with R the (empty) root of the tree-decomposition, $c(R; ;)$ is indeed the maximum possible number of realizable edges when imposing a $(tw\ tw^0)$-diet to T. The recurrence relation of Theorem 6 therefore lends itself to a dynamic programming approach, over the tree-decomposition T following leaf-to-root order, for the problem.

It is reasonable to assume the number of bags in a tree decomposition to be linear in n (this is for instance the case for anice tree decomposition [42, 29], or for a tree decomposition obtained from anelimination ordering, see [43, 17]). Therefore, the number of entries to the table is $O(3^{tw}\ n)$, given that a bag X may be colored in $3^{jX\ j}$ ways, and that the maximum size of X is $tw + 1$. For a given entry X_i, one must rst enumerate all possible choices of $m : f\ ^1(o) !\ [1:::]$, map assigning one child of X_i to each orange vertex in X_i. There are $O(\ ^{tw+1})$ possibilities for m in the worst case, as $jf\ ^1(o)j\ \ tw + 1$. Then, for each child Y_j, one must enumerate all possible colorings f_j^0 compatible with f. Possibilities for $f_j^0(u)$ depend on the color by f :

- if $u \in X_i\ !\ f_j^0(u) =$ o or g
- if $f(u) =$ g $!\ f_j^0(u) =$ g or r
- if $f(u) =$ o $!\ f_j^0(u) =$ o or g if $m[u] = j$ or $f_j^0(u) =$ r otherwise.
- if $f(u) =$ r $!\ f_j^0(u) =$ r

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

Corollary 1 Binary-Tree-Diet (= 2) admits an FPT algorithm for the tw parameter.

A pseudo-code implementation of the algorithm, using memoization, is included in Supplementary Section B

4.2 For path decompositions

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

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

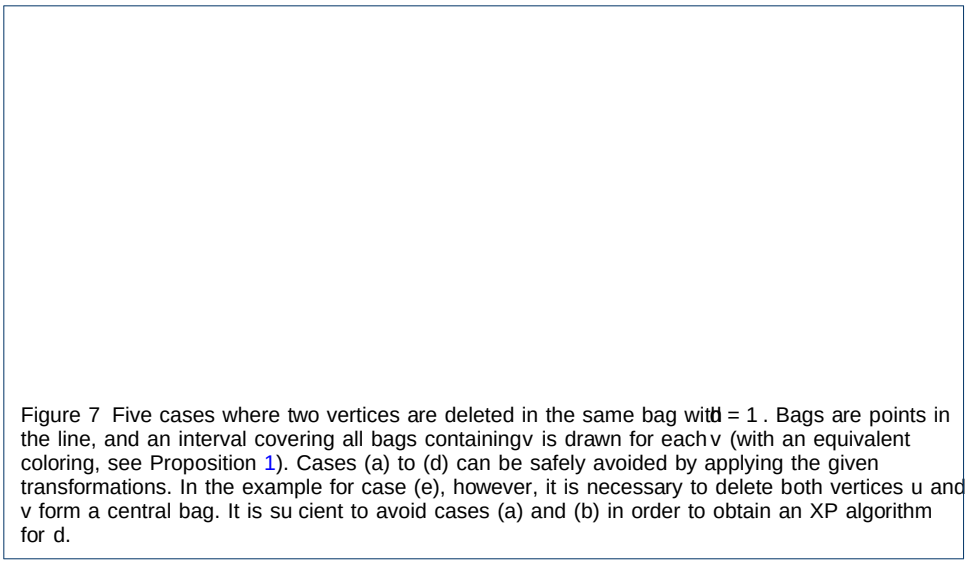

Figure 7 Five cases where two vertices are deleted in the same bag with $d = 1$. Bags are points in the line, and an interval covering all bags containing v is drawn for each v (with an equivalent coloring, see Proposition 1). Cases (a) to (d) can be safely avoided by applying the given transformations. 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.

Proof of Proposition 2 Consider such a coloring C with a maximal number of green vertices. We show that it is d-simple. Assume the path-decomposition T is rooted in bag X_1 and each X_i is the parent of X_{i+1}. Pick i to be the smallest index so that at least $d+1$ vertices in X_i are colored red by C, assume any such i exists. Then one of these vertices, say u, is not colored red in X_{i-1} (either because $i = 1$, or it is not in X_{i-1}, or it is orange or green in X_{i-1}). Consider C' obtained by C and coloring u green in X_i. Then C' satisfies local rules R1 through R4 (a green vertex may be absent, green or orange in the parent bag, and a red vertex may be green in the parent bag). Furthermore, it is d-diet-valid since it still removes at least d (red) vertices in X_i. Overall C' is another d-diet-valid coloring with more green vertices: a contradiction, so no such i exist (and no bag has $d+1$ red vertices). The same argument works symmetrically for orange vertices. Overall, C is d-simple. □

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

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

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 7 and Supplementary Section B. Its main primitives are made available for Python scripting through pybind11 [44].

It actually allows to solve a generalized weighted version of Tree Diet, as explained in Supplementary Section B. This feature allows to favour the conservation

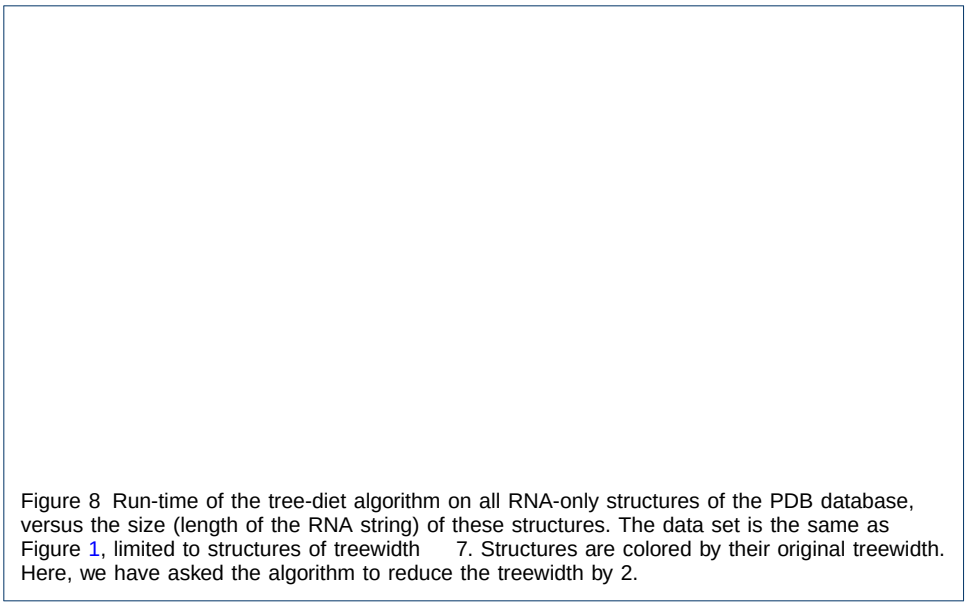

Figure 8 Run-time of the tree-diet algorithm on all RNA-only structures of the PDB database, versus the size (length of the RNA string) of these structures. The data set is the same as Figure 1, limited to structures of treewidth 7. Structures are colored by their original treewidth. Here, we have asked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

of important edges (e.g. RNA backbone) during simplification, by assigning them a much larger weight compared to other edges. Our implementation is freely available at https://gitlab.inria.fr/amibio/tree-diet.

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

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

This discrepancy in the complexities/runtimes of the preprocessing and sampling suggests an alternative strategy: relaxing the set of constraints to $(S^0; D^0)$, with $(S^0[D^0) \supseteq (S[D)$, and compensating it through a rejection of sequences violating constraints in $(S;D) \setminus (S^0;D^0)$. The relaxed algorithm would remain unbiased, while the average-case time complexity of the rejection algorithm would be in $(k \cdot q \cdot (n+m))$ time, where q represents the relative increase of the partition function (the sequence space) induced by the relaxation. The preprocessing step would retain

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 9 (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.

the same complexity, but based on a (reduced) treewidth $tw' \leq tw$ for the relaxed graph $G' = ([1;n]; S'[D')$.

These complexities enable a tradeoff between the rejection (time), and the preprocessing (space), which may be critical to unlock future applications of RNA design. Indeed, the treewidth can be decreased by removing relatively few base pairs, as demonstrated below using our algorithm on pairs inferred for hard design instances.

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

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

Rough estimates can be provided for the tradeoff between the rejection and preprocessing complexities, by assuming that removing a DBP homogeneously increases the value of Z by a factor $\gamma := 16/10$ (#pairs/#incomp. pairs). The relative increase in partition function is then γ^b, when b base pairs are removed. For EteRNA22, reducing the treewidth by 2 units (6 → 4), i.e. a 16 fold reduction of the memory and preprocessing time, can be achieved by removing 9 DBPs, i.e. a 69 fold expected increase in the time of the generation phase. For EteRNA77, the same

16 fold ($tw' = 9 \rightarrow 7$) reduction of the preprocessing time/space can be achieved through an estimated 4 fold increase of the generation time. A more aggressive 256 fold memory gain can be achieved at the expense of an estimated 1 152 fold increase in generation time. Given the large typical asymmetry in runtimes and implementation constants between the computation-heavy preprocessing and, relatively light, generation phases, the availability of an algorithm for the tree-diet problem provides new options, especially to circumvent memory limitations.

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 [13], considering RNA base pairs while aligning distant homologs. In the presence of complex structural features (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. [12] describe a $(n{:}m^{tw+1})$ algorithm for optimally aligning a structured RNA of length n onto a genomic region of length m. It optimizes an alignment score that includes: i) substitution costs for matches/mismatches of individual nucleotides and base pairs (including arc-breaking) based on the RIBOSUM matrices [45]; and ii) an affine gap cost model [46]. We used the implementation of the Rinaudo et al. algorithm, implemented in the LicoRNA software package [47, 48].

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 [49, 50]. A 2.8Å resolution 3D model of the c-di-GMP-II riboswitch in *C. acetobutylicum*, proposed by Smith et al. [51] based on X-ray crystallography, was retrieved from the PDB [24] (PDBID: 3Q3Z). We annotated its base pairs geometrically using the DSSR method [52]. The canonical base pairs, supplemented with the backbone connections, were then accumulated in a graph, for which we heuristically computed an initial tree decomposition T_4, having treewidth $tw = 4$.

We simplified our the initial tree decomposition T_4, and obtained simplified models T_3; and 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; T_{2D}) and sequence (T_{1D}). We used LicoRNA to predict an alignment $a_{T,w}$ of each original/simplified tree decomposition T onto each sequence w of the c-di-GMP-II riboswitch family in the RFAM database [13] (RF01786). Finally, we reported the LicoRNA runtime, and computed the Sum of Pairs Score (SPS) [53] as a measure of the accuracy of $a_{T,w}$ against a reference alignment $a^?_w$:

$$\mathrm{SPS}(a_{T,w}; a^?_w) = \frac{|\,\mathrm{MatchedCols}(a_{T,w}) \setminus \mathrm{MatchedCols}(a^?_w)\,|}{|\,\mathrm{MatchedCols}(a^?_w)\,|},$$

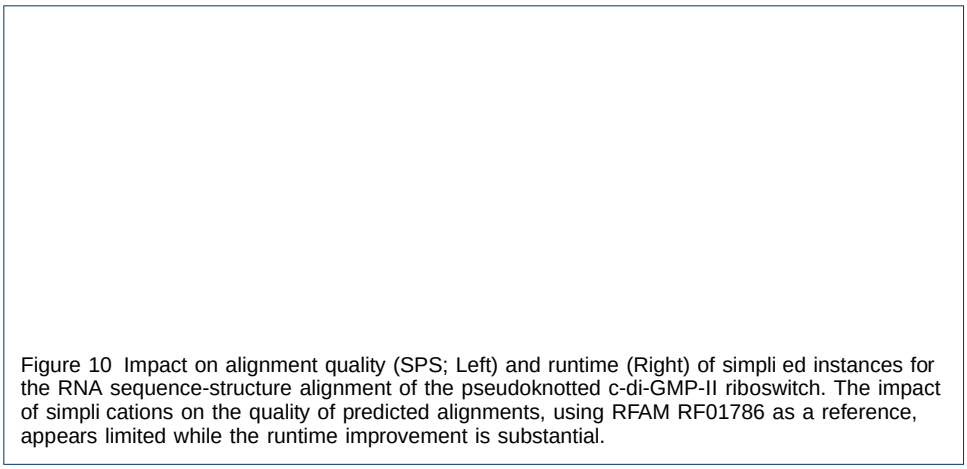

Figure 10 Impact on alignment quality (SPS; Left) and runtime (Right) of simplified instances for the RNA sequence-structure alignment of the pseudoknotted c-di-GMP-II riboswitch. The impact of simplifications on the quality of predicted alignments, using RFAM RF01786 as a reference, appears limited while the runtime improvement is substantial.

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

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

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 λn; $\lambda > 1$. Classically, one would align S against individual windows, and report those associated with an admissible alignment cost. This strategy would have an overall $(g \cdot n^{tw+2})$ time complexity, applying for instance the algorithm of [12].

Our instance simplification framework enables an alternative strategy, that incrementally filters out unsuitable windows based on models of increasing granularity. Indeed, for any given target sequence, the min alignment cost c obtained for a simplified instance of treewidth tw can be corrected (cf Supplementary Section D) into a lower bound $c^?$ for the min alignment cost $c^?_0$ of the full-treewidth instance tw. Any window such that $c^? > \varepsilon$ thus also obeys $c^?_0 > \varepsilon$, 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

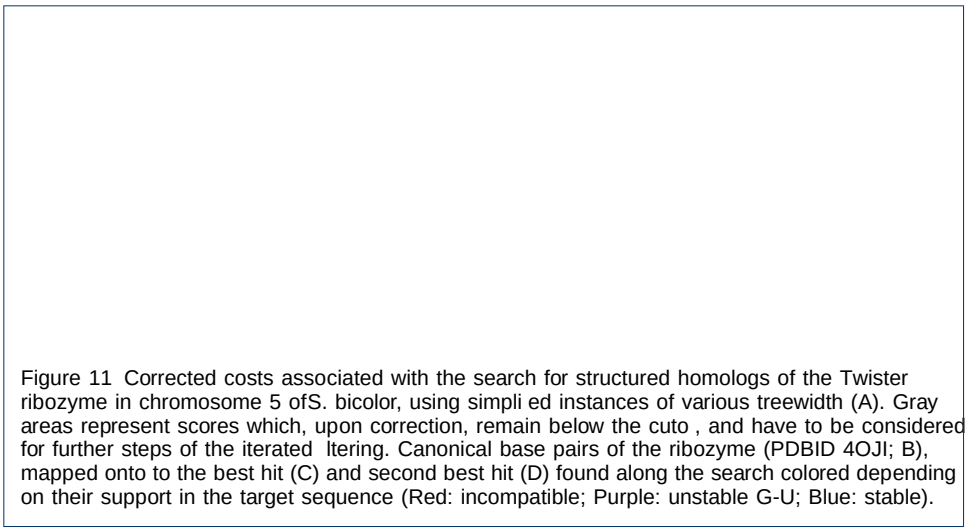

Figure 11 Corrected costs associated with the search for structured homologs of the Twister ribozyme in chromosome 5 ofS. bicolor, using simpli ed instances of various treewidth (A). Gray areas represent scores which, upon correction, remain below the cuto , and have to be considered for further steps of the iterated ltering. 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).

for increasing treewidth values (see Figure 10-right) this strategy is expected to yield substantial runtime savings.

We used this strategy to search occurrences of the Twister ribozyme (PDBID 4OJI), a highly-structured (tw = 5) 54nts RNA initially found in O. sativa (Asian rice) [54]. We targeted the S. bicolor genome (sorghum), focusing on a 10kb region centered on the 2,485,140 position of the 5th chromosome, where an instance of the ribozyme was suspected within an uncharacterized transcript (LOC110435504). The 4OJI sequence and structure were extracted from the 3D model as above, and included into a tree decomposition T_5 (73 edges), simpli ed into T_4 (71 edges),T_3 (68 edges) andT_2 (61 edges) using the tree-diet algorithm.

We aligned all tree decompositions against all windows of size 58nts using a 13nts o set, and measured the score and runtime of the iterative ltering strategy using a cost cuto " = 5. The search recovers the suspected occurrence of twister as its best result (Figure 11.C), but produced hits (cf Figure 11.D) with comparable sequence conservation that could be the object of further studies. Regarding the ltering strategy, while T_2 only allows to rule out 3 windows out of 769,T_3 allows to eliminate an important proportion of putative targets, retaining only 109 windows, further reduced to 15 windows by T_4, 6 of which end up as nal hits for the full model T_5 (cf Figure 11.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 ltering can be safely achieved based on simpli ed input graphs.

6.1 Open questions

Regarding the parameterized complexity of Graph-Diet and Tree-Diet, some questions remain open (see Table 1): an FPT algorithm for Tree-Diet (ideally, with $2^{O(tw)}n$ running time), would be the most desirable, if possible satisfying the backbone constraints. The existence of such an algorithm is not trivial. In particular, it is perhaps worth noting that it is not implied by the existence of an FPT algorithm for graph-diet with the input treewidth as a parameter (1). Indeed, in comparison to the latter, tree-diet subtly restricts the search space to tree decompositions that are subsets of the input tree decomposition. It follows that the result of graph diet for a graph G may substantially differ from the result of tree-diet given a tree decomposition T of G as input. We also aim at trying to give efficient exact algorithms for graph diet in the context of RNA (we conjecture this is impossible in the general case). Finally, we did not include the number of deleted edges in our multivariate analysis: even though in practice it is more difficult a priori to guarantee their small number, we expect it can be used to improve the running time in many cases.

6.2 Backbone Preservation.

In two of our applications, the RNA secondary structure graph contains two types of edges: those representing the *backbone* of the sequence (i.e., between consecutive bases) and those representing base pair bonds. In practice, we want all backbone edges to be visible in the resulting tree-decomposition, and only base pairs may be lost. This can be integrated to the Tree-Diet model (and to our algorithms) using weighted edges, using the total weight rather than the count of deleted edges for the objective function. Note that some instances might be unrealizable (with no tree-diet preserving the backbone, especially for low tw^0). In most cases, ad-hoc bag duplications can help avoid this issue. The design of pre-processing methods, involving bag duplications or other operations on tree decompositions, and aimed at ensuring the existence of a backbone-preserving tree-diet will be the subject of future work.

From a theoretical perspective, weighted edges may only increase the algorithmic complexity of the problems. However, a more precise model could consider graphs which already include a hamiltonian path (the backbone), and the remaining edges form a degree-one or two subgraph. Such extra properties may, in some cases, actually reduce the complexity of the problem. As an extreme case, we conjecture the Path-Diet problem for $tw^0 = 1$ becomes polynomial in this setting.

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.

Availability of data and materials
Source code of tree-diet method available at: https://gitlab.inria.fr/amibio/tree-diet

Competing interests
The authors declare that they have no competing interests.

Author details
[1]LIX CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France. [2]LIGM, CNRS, Univ Gustave Eiffel, F77454 Marne-la-Vallée, France.

References

1. Weller, M., Chateau, A., Giroudeau, R.: Exact approaches for sca olding. BMC Bioinformatics 16(S14) (2015). doi:10.1186/1471-2105-16-s14-s2
2. Xu, J.: Rapid protein side-chain packing via tree decomposition. In: Research in Computational Molecular Biology (RECOMB 2005). Lecture Notes in Computer Science, vol. 3500, pp. 423{439. Springer, Cambridge, USA (2005). doi:10.1007/11415770_32
3. Bulteau, L., Fertin, G., Jiang, M., Rusu, I.: Tractability and approximability of maximal strip recovery. Theoretical Computer Science 440, 14{28 (2012)
4. Baste, J., Paul, C., Sau, I., Scornavacca, C.: E cient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees. Bulletin of Mathematical Biology 79(4), 920{938 (2017). doi:10.1007/s11538-017-0260-y
5. Bulteau, L., Weller, M.: Parameterized algorithms in bioinformatics: An overview. Algorithms 12(12), 256 (2019). doi:10.3390/a12120256
6. Waterman, M.S.: Secondary structure of single stranded nucleic acids. Advances in Mathematics Supplementary Studies 1(1), 167{212 (1978)
7. Xayaphoummine, A., Bucher, T., Thalmann, F., Isambert, H.: Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations. Proc. Natl. Acad. Sci. U. S. A. 100(26), 15310{15315 (2003)
8. Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discrete Appl. Math. 104(1-3), 45{62 (2000). doi:10.1016/S0166-218X(00)00186-4
9. Lyngs , R.B., Pedersen, C.N.S.: RNA pseudoknot prediction in energy-based models. Journal of Computational Biology 7(3-4), 409{427 (2000)
10. Sheikh, S., Backofen, R., Ponty, Y.: Impact Of The Energy Model On The Complexity Of RNA Folding With Pseudoknots. In: Kärkkäinen, J., Stoye, J. (eds.) CPM - 23rd Annual Symposium on Combinatorial Pattern Matching. Combinatorial Pattern Matching, vol. 7354, pp. 321{333. Springer, Helsinki, Finland (2012). doi:10.1007/978-3-642-31265-6_26. Juha Kärkkäinen
11. Blin, G., Denise, A., Dulucq, S., Herrbach, C., Touzet, H.: Alignments of RNA structures. IEEE/ACM Transactions on Computational Biology and Bioinformatics 7(2), 309{322 (2010). doi:10.1109/tcbb.2008.28
12. Rinaudo, P., Ponty, Y., Barth, D., Denise, A.: Tree decomposition and parameterized algorithms for rna structure-sequence alignment including tertiary interactions and pseudoknots. In: Raphael, B., Tang, J. (eds.) Algorithms in Bioinformatics, pp. 149{164. Springer, Ljubljana, Slovenia (2012)
13. Kalvari, I., Nawrocki, E.P., Ontiveros-Palacios, N., Argasinska, J., Lamkiewicz, K., Marz, M., Gri ths-Jones, S., To ano-Nioche, C., Gautheret, D., Weinberg, Z., Rivas, E., Eddy, S.R., Finn, R.D., Bateman, A., Petrov, A.I.: Rfam 14: expanded coverage of metagenomic, viral and microRNA families. Nucleic Acids Research 49(D1), 192{200 (2020). doi:10.1093/nar/gkaa1047
14. Sarrazin-Gendron, R., Yao, H.-T., Reinharz, V., Oliver, C.G., Ponty, Y., Waldispühl, J.: Stochastic sampling of structural contexts improves the scalability and accuracy of RNA 3d module identi cation. In: Lecture Notes in Computer Science, pp. 186{201. Springer, Padua, Italy (2020). doi:10.1007/978-3-030-45257-5_12
15. Leontis, N.B., Westhof, E.: Geometric nomenclature and classi cation of RNA base pairs. RNA 7(4), 499{512 (2001)
16. Reinharz, V., Soulé, A., Westhof, E., Waldispühl, J., Denise, A.: Mining for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network families. Nucleic Acids Research 46(8), 3841{3851 (2018). doi:10.1093/nar/gky197
17. Gogate, V., Dechter, R.: A complete anytime algorithm for treewidth. arXiv preprint arXiv:1207.4109 (2012)
18. Bodlaender, H.L., Koster, A.M.: Treewidth computations i. upper bounds. Information and Computation 208(3), 259{275 (2010)
19. Song, Y., Liu, C., Malmberg, R., Pan, F., Cai, L.: Tree decomposition based fast search of RNA structures including pseudoknots in genomes. In: Computational Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE, pp. 223{234 (2005). IEEE
20. Han, B., Dost, B., Bafna, V., Zhang, S.: Structural alignment of pseudoknotted RNA. Journal of Computational Biology 15(5), 489{504 (2008). doi:10.1089/cmb.2007.0214
21. Vucinic, J., Simoncini, D., Ru ni, M., Barbe, S., Schiex, T.: Positive multistate protein design. Bioinformatics 36(1), 122{130 (2019). doi:10.1093/bioinformatics/btz497
22. Yao, H.-T., Waldispühl, J., Ponty, Y., Will, S.: Taming Disruptive Base Pairs to Reconcile Positive and Negative Structural Design of RNA. In: Research in Computational Molecular Biology - 25th International Conference on Research in Computational Molecular Biology (RECOMB 2021), Padova, France (2021)
23. Hammer, S., Wang, W., Will, S., Ponty, Y.: Fixed-parameter tractable sampling for RNA design with multiple target structures. BMC Bioinformatics 20(1) (2019). doi:10.1186/s12859-019-2784-7
24. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The protein data bank. Nucleic acids research 28, 235{242 (2000). doi:10.1093/nar/28.1.235
25. Lu, X.-J., Bussemaker, H.J., Olson, W.K.: Dssr: an integrated software tool for dissecting the spatial structure of rna. Nucleic acids research 43(21), 142{142 (2015)
26. van Dijk, T., van den Heuvel, J.-P., Slob, W.: Computing treewidth with libtw. Citeseer. http://citeseerx. ist. psu. edu/viewdoc/download (2006)
27. Bodlaender, H.L.: A linear-time algorithm for nding tree-decompositions of small treewidth. SIAM Journal on computing 25(6), 1305{1317 (1996)
28. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Berlin, Heiderlberg (2012)
29. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms vol. 5. Springer, Cham (2015)
30. El-Mallah, E.S., Colbourn, C.J.: The complexity of some edge deletion problems. IEEE transactions on circuits and systems 35(3), 354{362 (1988)
31. Crespelle, C., Drange, P.G., Fomin, F.V., Golovach, P.A.: A survey of parameterized algorithms and the

complexity of edge modification. arXiv preprint arXiv:2001.06867 (2020)
32. Cai, L.: Parameterized complexity of vertex colouring. Discrete Applied Mathematics 127(3), 415–429 (2003)
33. Lovász, L.: Graph minor theory. Bulletin of the American Mathematical Society 43(1), 75–86 (2006)
34. Robertson, N., Seymour, P.D.: Graph minors. xiii. the disjoint paths problem. Journal of combinatorial theory, Series B 63(1), 65–110 (1995)
35. Cygan, M., Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: On the hardness of losing width. In: International Symposium on Parameterized and Exact Computation, pp. 159–168 (2011). Springer
36. Baste, J., Sau, I., Thilikos, D.M.: Hitting minors on bounded treewidth graphs. i. general upper bounds. SIAM J. Discret. Math. 34(3), 1623–1648 (2020). doi:10.1137/19M1287146
37. Courcelle, B.: The monadic second-order logic of graphs iii: Tree-decompositions, minors and complexity issues. RAIRO-Theoretical Informatics and Applications-Informatique Theorique et Applications 26(3), 257–286 (1992)
38. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. Journal of Algorithms 12(2), 308–340 (1991)
39. Saitoh, T., Yoshinaka, R., Bodlaender, H.L.: Fixed-treewidth-efficient algorithms for edge-deletion to interval graph classes. In: Algorithms and Computation - 15th International Conference and Workshops (WALCOM 2021). Lecture Notes in Computer Science, vol. 12635, pp. 142–153. Springer, Yangon, Myanmar (2021). doi:10.1007/978-3-030-68211-8_12. https://doi.org/10.1007/978-3-030-68211-8_12
40. Tan, J., Zhang, L.: The consecutive ones submatrix problem for sparse matrices. Algorithmica 48(3), 287–299 (2007)
41. Proskurowski, A., Telle, J.A.: Classes of graphs with restricted interval models. Discrete Mathematics & Theoretical Computer Science 3(4) (2006)
42. Bodlaender, H.L., Koster, A.M.: Combinatorial optimization on graphs of bounded treewidth. The Computer Journal 51(3), 255–269 (2008)
43. Bodlaender, H.L.: Discovering treewidth. In: International Conference on Current Trends in Theory and Practice of Computer Science, pp. 1–16 (2005). Springer
44. Jakob, W., Rhinelander, J., Moldovan, D.: pybind11 – Seamless operability between C++11 and Python. https://github.com/pybind/pybind11 (2017)
45. Klein, R.J., Eddy, S.R.: Rsearch: finding homologs of single structured RNA sequences. BMC bioinformatics 4(1), 44 (2003)
46. Rivas, E., Eddy, S.R.: Parameterizing sequence alignment with an explicit evolutionary model. BMC bioinformatics 16(1), 406 (2015)
47. Wang, W.: Practical sequence-structure alignment of rnas with pseudoknots. PhD thesis, Universite Paris-Saclay, School of Computer Science (2017)
48. Wang, W., Denise, A., Ponty, Y.: LicoRNA: aLignment of Complex RNAs v1.0 (2017). https://licorna.lri.fr
49. Sudarsan, N., Lee, E.R., Weinberg, Z., Moy, R.H., Kim, J.N., Link, K.H., Breaker, R.R.: Riboswitches in eubacteria sense the second messenger cyclic di-gmp. Science 321(5887), 411–413 (2008). doi:10.1126/science.1159519. https://science.sciencemag.org/content/321/5887/411.full.pdf
50. Tamayo, R.: Cyclic diguanylate riboswitches control bacterial pathogenesis mechanisms. PLOS Pathogens 15(2), 1–7 (2019). doi:10.1371/journal.ppat.1007529
51. Smith, K.D., Shanahan, C.A., Moore, E.L., Simon, A.C., Strobel, S.A.: 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). doi:10.1073/pnas.1018857108. https://www.pnas.org/content/108/19/7757.full.pdf
52. Lu, X.-J., Bussemaker, H.J., Olson, W.K.: DSSR: an integrated software tool for dissecting the spatial structure of RNA. Nucleic Acids Research 43(21), 142–142 (2015). doi:10.1093/nar/gkv716. https://academic.oup.com/nar/article-pdf/43/21/e142/17435026/gkv716.pdf
53. Thompson, J.D., Plewniak, F., Poch, O.: BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics 15(1), 87–88 (1999). doi:10.1093/bioinformatics/15.1.87. https://academic.oup.com/bioinformatics/article-pdf/15/1/87/9731974/150087.pdf
54. Liu, Y., Wilson, T.J., McPhee, S.A., Lilley, D.M.: Crystal structure and mechanistic investigation of the twister ribozyme. Nature chemical biology 10(9), 739–744 (2014)

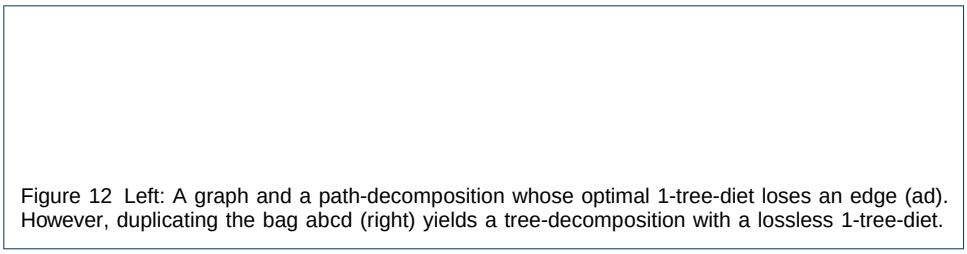

Figure 12 Left: A graph and a path-decomposition whose optimal 1-tree-diet loses an edge (ad). However, duplicating the bag abcd (right) yields a tree-decomposition with a lossless 1-tree-diet.

Supplementary Material

Section A: Editing Trees before the Diet

Any tree decomposition can be transformed into a binary one through the duplications of bags having more than 2 children. To do so in practice, one will, as long as the tree decomposition is not binary, apply the following transformation:

1. Find a bag X with children $Y_1, \ldots, Y_\ell$ and $\ell > 2$.
2. Introduce a new bag X' with the same content as X and locally modify the tree decomposition in the following way: X will now have Y_1 and X' as children, while X' will have $Y_2 \ldots Y_\ell$.

When it is no longer possible to apply this transformation, the tree decomposition is binary. For each bag having originally $\ell > 2$ children in the decomposition, $\ell - 1$ new bags have been introduced. In total, with N_{bags} the original number of bags in the decomposition, strictly less than N_{bags} new bags have been introduced (each new bag is associated to an edge of the original tree decomposition).

This tranformation is in fact the first step towards obtaining a nice tree decomposition [42, 29].

A question that arises then is what impact these modifications may have on the output of Tree-Diet, when applied to the tree decomposition given as input. We argue that duplication operations (as used above to get a binary tree decomposition) can only improve the solution, i.e decrease the number of lost edges. Indeed, within the coloring formulation of the problem, new bags yield new opportunities for an edge to be represented, with both its end-points green in some bag. See Figure 12 for an illustration.

More generally, any operation on the input tree decomposition that does not suppress any of the original bags can only improve the solution to the Tree Diet problem. We do not tackle here the problem of finding the best edition operations to apply onto a tree decomposition given as input to Tree Diet, which is an a priori difficult task.

Section B: 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 [44] 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 Algorithm 1, 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.

```
Input          :  Tree-decomposition 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 T
Side-Product:  A lled table c[X_i; f], 8X_i bag and f coloring of X_i
Function optim_numreal_edges(X_i; f; G; tw'; W):
    if c[X_i; f] already computed then return c[X_i; f]; ;
    if jf^1(o) [ f^1(r)j (jX_ij tw' 1) then
        //not enough removals. ;
        c[X_i; f] = 1 ;
        return c[X_i; f];
    end
    if X_i == leaf then
        c[X_i; f] = 0 ;
        return c[X_i; f];
    end
    int ans = 1 ;
    for m 2 orange_maps(X_i; f) do
        int ans_m = 0;
        for Y_j 2 X_i:children do
            int ans_j = 1 ;
            for f'_j 2 compatible(f; m; X_i; Y_j) do
                int val = 0;
                val += count(f; f'_j; W);
                val += optim_numreal_edges(Y_j; f'_j; G; tw');
                if val ans_j then ans_j = val;
                ;
            end
            ans_m += ans_j;
        end
        if ans_m ans then ans = ans_m; ;
    end
    c[X_i; f] = ans
    return c[X_i; f];
end
```

Algorithm 1: Dynamic programming algorithm for Tree-Diet .

Section C: Correctness of the rejection-based sampling of RNA designs

A recent method for RNA design, called RNAPond [22], 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, identi es 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 D of DBPs, and excluded in subsequent generations through an assignment of non-binding pairs of nucleotides, outside of $B := \{(G;C);(C;G);(A;U);(U;A);(G;U);(U;G)\}$:

```
Input  :  Tree-decomposition T , graph G, target width tw', table c, edge weights W
Output :  Optimal (tw  tw')-diet-valid coloration for T
Function optim_coloring( X_i ; f; G; tw'; c) :
    if X_i == leaf then
        return ; ;
    end
    coloring C = ; ;
    for m 2 orange_maps(X_i ; f ) do
        int ans_m = 0;
        coloring best_fjs = [];
        for Y_j 2 X_i :children do
            int best_valj =  1  ;
            int best_fj = ; ;
            for f_j' 2 compatible (f; m; X_i ; Y_j ) do
                int val = 0;
                val +=  count (f; f_j'; W );
                val +=  c[Y_j ; f_j'];
                if val  best_valj then
                    best_valj = val;
                    best_fj = f_j';
                end
            end
            ans_m += best _valj;
            best_fjs.add(best_fj);
        end
        if ans_m ==  c[X_i ; f ] then
            C+ = [ f_j' for f_j' in best_fjs ];
            C+ = [ optim _coloring( Y_j ; f_j'; G; tw'; c)  for f_j' in best_fjs ];
            break;                                                  // break loop over m
        end
    end
    return C;
end
```

Algorithm 2: Backtracking procedure for Tree-Diet .

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

$$\forall w \in W_{S;D} : P(w \mid D; S) = \frac{e^{-E_{w;S}}}{Z_{S;D}} \quad \text{with} \quad Z_{S;D} := \sum_{w' \in W_{S;D}} e^{-E_{w';S}} \tag{1}$$

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

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 $(S'; D')$ of the initial constraints ($S' \cup D' \supseteq S \cup D$), and iterates Yao et al.'s algorithm to generate sequences in $W_{S';D'} \supseteq W_{S;D}$, rejecting those outside of $W_{S;D}$, until k suitable ones are obtained. The rejection algorithm generates a given sequence $w \in W_{S;D}$ on its first attempt with probability $p := e^{-E_{w;S}}/Z_{S';D'}$ and, more generally, after r rejections with probability $(1 - q)^r p$ with $q := Z_{S;D}/Z_{S';D'}$. The overall probability of emitting w is thus

$$p \sum_{r \geq 0} (1 - q)^r = \frac{p}{q} = \frac{e^{-E_{w;S}}}{Z_{S;D}} = P(w \mid D; S).$$

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

Meanwhile, the average-case complexity can be impacted by the strategy. Indeed, the relaxed instance $(S'; D')$ 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 $q := Z_{S';D'}/Z_{S;D}$, 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 $(n \cdot |D'| \cdot 4^{tw'} + k \cdot q \cdot (n + |D'|))$ time and $(n \cdot 4^{tw'})$ 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.

Section D: 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, then there is no need to align w to the full model S, as the resulting cost is guaranteed to stay above the selection cutoff.

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[1]. We consider the following cost function, adapted from eciteRinaudo2012, 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 $b = \perp$);
- $\phi(a, b, c, d)$ returns 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 3

$$c' - \sum_{\substack{i \text{ unpaired in } S^0, \\ \text{paired in } S}} \max_b \gamma(S_i, b) + \sum_{(i,j)\in S\setminus S^0} \min_{a,b} \phi(S_i, S_j, a, b) \leq c^\star \tag{2}$$

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

$$C(S, w, \mu) = C(S', w, \mu) - \sum_{\substack{i \text{ unpaired in } S^0, \\ \text{paired in } S, \\ \text{and } k:=\mu_i}} \gamma(S_i, w_k) + \sum_{\substack{(i,j)\in S\setminus S^0 \\ \text{s.t. } (k,l):=(\mu_i,\mu_j)}} \phi(S_i, S_j, w_k, w_l).$$

Minimizing over all alignment μ, one obtains

$$\min_{\mu} C(S, w, \mu) = \min_{\mu} C(S', w, \mu) - \sum_{\substack{i \text{ unpaired in } S^0, \\ \text{paired in } S, \\ \text{and } k:=\mu_i}} \gamma(S_i, w_k) + \sum_{\substack{(i,j)\in S\setminus S^0 \\ \text{s.t. } (k,l):=(\mu_i,\mu_j)}} \phi(S_i, S_j, w_k, w_l).$$

[1] 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 in this discussion for the sake of simplicity.