



HAL
open science

Automated design of dynamic programming schemes for RNA folding with pseudoknots

Bertrand Marchand, Sebastian Will, Sarah Berkemer, Laurent Bulteau, Yann Ponty

► **To cite this version:**

Bertrand Marchand, Sebastian Will, Sarah Berkemer, Laurent Bulteau, Yann Ponty. Automated design of dynamic programming schemes for RNA folding with pseudoknots. WABI 2022 - 22nd Workshop on Algorithms in Bioinformatics, Sep 2022, Potsdam, Germany. hal-03676377v1

HAL Id: hal-03676377

<https://inria.hal.science/hal-03676377v1>

Submitted on 23 May 2022 (v1), last revised 11 Aug 2022 (v2)

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

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



Distributed under a Creative Commons Attribution 4.0 International License

Automated design of dynamic programming schemes for RNA folding with pseudoknots

Bertrand Marchand  

LIX (UMR 7161), Ecole Polytechnique, Institut Polytechnique de Paris, France

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

Sebastian Will  

LIX (UMR 7161), Ecole Polytechnique, Institut Polytechnique de Paris, France

Sarah J. Berkemer  

LIX (UMR 7161), Ecole Polytechnique, Institut Polytechnique de Paris, France

Laurent Bulteau  

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

Yann Ponty¹  

LIX (UMR 7161), Ecole Polytechnique, Institut Polytechnique de Paris, France

Abstract

Despite being a textbook application of dynamic programming (DP) and routine task in RNA structure analysis, RNA secondary structure prediction remains challenging whenever pseudoknots come into play. To circumvent the NP-hardness of energy minimization in realistic energy models, specialized algorithms have been proposed for restricted conformation classes that capture the most frequently observed configurations.

While these methods rely on hand-crafted DP schemes, we generalize and fully automatize the design of DP pseudoknot prediction algorithms. We formalize the problem of designing DP algorithms for an (infinite) class of conformations, modeled by (a finite number of) fatgraphs, and automatically build DP schemes minimizing their algorithmic complexity. We propose an algorithm for the problem, based on the tree-decomposition of a well-chosen representative structure, which we simplify and reinterpret as a DP scheme. The algorithm is fixed-parameter tractable for the tree-width tw of the fatgraph, and its output represents a $\mathcal{O}(n^{tw+1})$ algorithm for predicting the MFE folding of an RNA of length n .

Our general framework supports general energy models, partition function computations, recursive substructures and partial folding, and could pave the way for algebraic dynamic programming beyond the context-free case.

2012 ACM Subject Classification ; Applied computing → Computational biology; Theory of computation → Dynamic programming

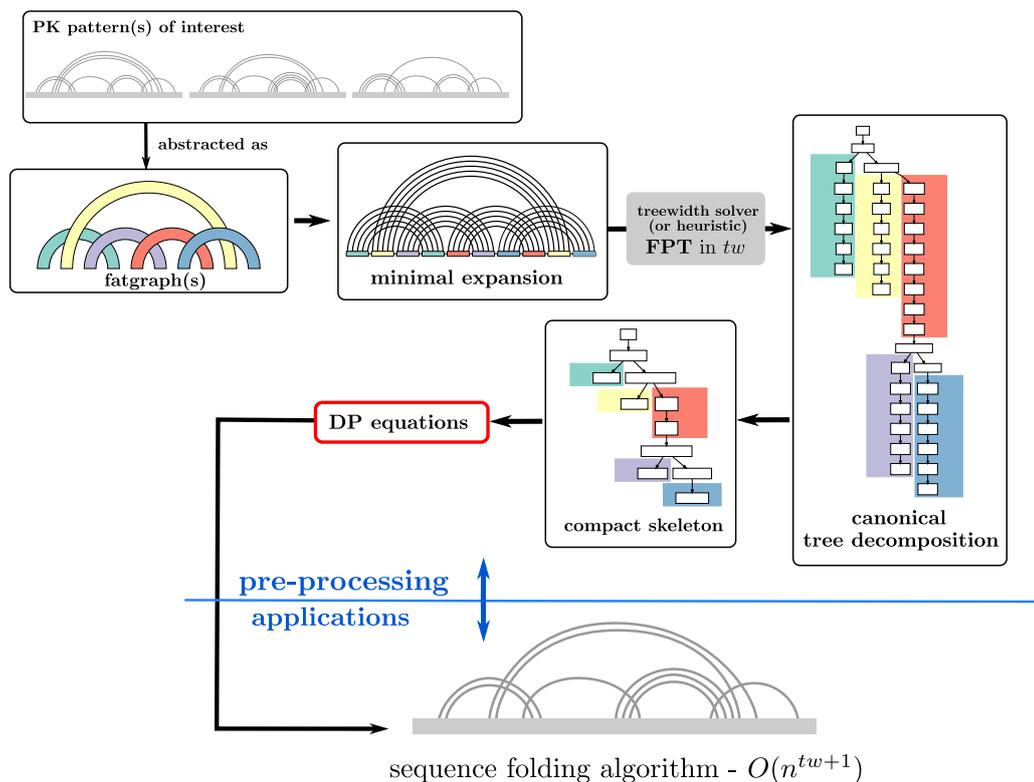
Keywords and phrases RNA folding, treewidth, dynamic programming

Digital Object Identifier 10.4230/LIPIcs.WABI.2022.23

Funding This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101029676.

¹ To whom correspondence should be addressed





■ **Figure 1** Given a finite number of arbitrary fatgraphs, a dynamic programming scheme for folding (restricted to the family of structures specified by the fatgraphs) is derived from canonical tree decompositions of minimal representative expansion of the helices, for each fatgraph. The workflow gives an overview of the steps of the algorithm whereas each step is described in more details in the subsequent sections and figures, hence, see Figure 2 for fatgraphs, Figure 8 and Section 3 for a detailed version of the canonical tree decomposition, Figure 5 for a detailed view of the compact skeleton of the tree decomposition.

37 1 Introduction

38 The function of non-coding RNAs is, to a large extent, determined by their structure. Struc-
 39 ture prediction algorithms therefore play a crucial role in (bio-)medical and pharmaceutical
 40 applications. The basis to determine more complex 3D structures of RNA molecules is
 41 set by accurately predicting the 2D or secondary structure of the RNA. There exist vari-
 42 ous RNA folding algorithms that predict an optimal secondary structure as *minimum free*
 43 *energy structure* of the given the RNA sequence in suitable thermodynamic models. In
 44 the most frequently used methods, this optimization is performed efficiently by a dynamic
 45 programming (DP) algorithm, e.g. `mfold` [46], `RNAfold` [23], `RNAstructure` [35]. A recent
 46 alternative to predictions based on experimentally determined energy parameters are machine
 47 learning approaches that train models on known secondary structures, e.g., `CONTRAFold` [15],
 48 `ContextFold` [45], `MXfold2` [39].

49 However, the most frequently used algorithms (including all of the above ones) optimize
 50 solely over pseudoknot-free structures [43], which don't contain crossing base pairs. Although
 51 pseudoknots appear in many RNA secondary structures, they have been omitted by initial

52 prediction algorithms due to their computational complexity [1], and the difficulty to score
 53 individual conformations [9]. Nevertheless, many algorithms have been proposed to predict
 54 at least certain pseudoknots. These methods are either based on exact DP algorithms such
 55 as `pknots-RE` [38], `NUPACK` [14], `gfold` [32], `Knotty` [21] or they use heuristics that don't
 56 guarantee exact solutions, e.g., `HotKnots` [34], `IPknot` [40, 39], `Hfold` [20].

57 Due to the hardness of PK prediction, efficient exact DP algorithms are necessarily
 58 restricted to certain categories of pseudoknotted structures. The underlying DP schemes
 59 are designed manually, guided by design to either i) support structures that are frequently
 60 observed in experimentally resolved structures (declarative categories); or ii) support the
 61 largest possible set of conformations, while remaining within a certain complexity (complexity-
 62 driven). For most categories, essentially declarative ones, there exists one or several helix
 63 arrangements, either observed in experimentally-determined structures or implicitly charac-
 64 terized by graph-theoretical properties (3 non-crossing [33], topologically bounded [32]) that
 65 need to be captured. A detailed overview of pseudoknot categories is given in [27]. Similar
 66 situations occur for RNA-RNA interactions [2], possibly including several RNA molecules.
 67 Interestingly, when more than two RNA strands are considered, existing algorithms restrict
 68 the joint conformation to crossing-free interactions [16], further motivating the design of
 69 algorithms beyond the case of pseudoknot-free secondary structures.

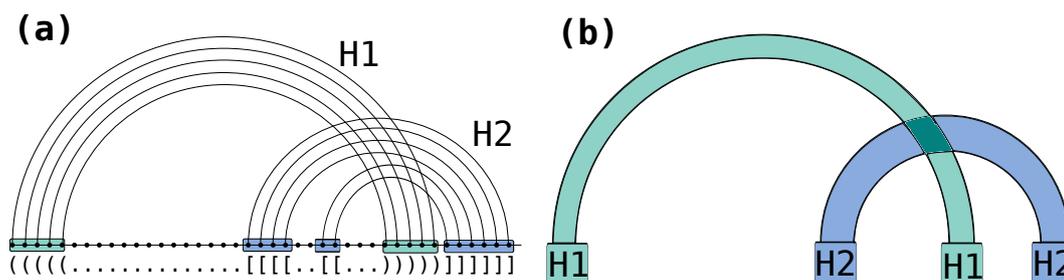
70 In this work, we described classes of pseudoknotted structures as fatgraphs [19, 32, 22, 29],
 71 an abstraction of RNA conformations related to RNA shapes [17] or shadows [33, 32]. We
 72 formalize the principles underlying the design of DP folding algorithms including pseudoknots,
 73 and, at the same time, give a formulation of the computational problem based on the design
 74 of DP algorithms. We show how to leverage tree-decompositions, computed on a minimal
 75 expansion of the input fatgraph, to derive DP schemes that use as little indices as possible,
 76 leading to an optimal complexity amongst DP schemes that obey a standard form. Our
 77 algorithm can be interpreted as a generalization of the algorithms underlying `LiCoRNA` [37]
 78 and `gfold` [32] and we propose a parameterized algorithm based on the treewidth (tw) of
 79 the underlying fatgraph.

80 In Section 2, we state our problem and define its input structure abstraction, the fatgraph.
 81 Then, we describe helix expansions of the fatgraph and its tree decompositions (Section 3).
 82 By minimal helix expansions and a derivation of the tree decomposition to its canonical
 83 form, we ensure the existence of an optimal DP scheme for the folding of pseudoknotted
 84 structures (Section 4). Figure 1 outlines the fundamental algorithm. Section 5, discusses
 85 extensions to combine multiple fatgraphs, include recursive substructure, and cover realistic
 86 energy models.

87 **2** Definitions and main result

88 We define an *RNA sequence* S as a word of length n over the nucleotides A, C, G and U ;
 89 moreover an *RNA secondary structure* (potentially, with pseudoknots) ω of S as a set of *base*
 90 *pairs* (i, j) between sequence positions i and j (in $1, \dots, n$), such that there is at most one base
 91 pair incident to each position. A *diagram* is a graph of nodes $1, \dots, n$ (the positions), connecting
 92 consecutive positions by directed edges $(i, i + 1)$ and moreover connecting positions by arcs,
 93 visualizing the *arc-annotation* of the sequence. Typically this is represented drawing the
 94 backbone linearly and the arcs on top. RNA secondary structures are naturally interpreted
 95 as diagrams.

96 One of our central concerns is the crossing configuration of arcs in a diagram. We define
 97 two arcs (i, j) and (i', j') in a diagram as *crossing* iff $i < i' < j < j'$ or $i' < i < j' < j$.



■ **Figure 2** (a) Diagram of a secondary structure with two crossing helices (H1 green, H2 blue). (b) fatgraph corresponding to the above structure such that helices are collapsed into bands and form the shadow of the structure.

98 Naturally, this leads to the notion of a conflict graph consisting of all the arcs of a diagram
 99 and connecting crossing arcs by a conflict edge.

100 Given a potentially conflicted set of base pairs, the associated *RNA structure graph* is the
 101 diagram consisting of one vertex per nucleotide, backbone link, and one arc per base pair.

102 A *fatgraph* [19, 32, 22, 29] is an abstraction of a family of pseudoknotted RNA structures
 103 displaying a specific conflict structure. It is typically represented as a *band diagram* (see
 104 Figure 1 and Figure 2), in which each band may represent a *helix* of arbitrary size, including
 105 bulges. An arc-annotation is said to be an *expansion* of a fatgraph if collapsing nested arcs
 106 and contracting isolated bases yields the band diagram of a fatgraph. Given a finite number
 107 of fatgraphs, we say a structure is a *recursive expansion* of these fatgraphs if decomposing the
 108 structure into conflict-connected components, collapsing nested arcs and contracting isolated
 109 bases only yields members of the given fatgraph set. For the purpose of this presentation
 110 (where we do not explicitly study structure topology), we moreover identify fatgraphs with
 111 their diagrams.

112 To make the connection to `gfold` [32] explicit, recursive expansions of fatgraphs are
 113 equivalently understood in terms of the shadows of a structure. The shadow of an RNA
 114 structure (or equivalently, its diagram) is defined in [32] as the diagram obtained by, firstly,
 115 removing all unpaired bases and non-crossing structures and, secondly, contracting all stacks
 116 (i.e. pairs of arcs between directly consecutive positions) to single arcs. Then, the class
 117 of recursive expansions of a set of input fatgraphs Γ is the class of structures, where the
 118 shadows of their conflict-connected components are in Γ .

119 In this paper, we consider a class of RNA folding problems in which the search space is
 120 restricted to recursive expansions of a user-specified finite set of fatgraphs. For the sake of
 121 simplicity, we first describe minimizing energy in a simple free-energy model \mathcal{E} , where the
 122 energy of a sequence/structure is obtained by summing the contributions of individual base
 123 pairs; moreover, we present the method initially without recursive substructure. Only later,
 124 in Section 5, we extend to the full problem in realistic energy models.

125 ► **Definition 1** ((Recursive) fatgraph folding problem).

126 **Input:** Finite collection of fatgraphs $\gamma_1, \dots, \gamma_p$, sequence S

127 **Output:** Minimum free energy arc-annotation for S according to \mathcal{E} , restricting the search to
 128 recursive expansions of the input fatgraphs.

129 Specifically, we solve the problem of automatic design of such pseudoknot prediction
 130 algorithms based on the defining set of fatgraphs.

131 ► **Definition 2** (Fatgraph folding algorithm design problem).

132 **Input:** Finite collection of fatgraphs $\gamma_1, \dots, \gamma_p$, sequence S

ALGORITHM 1 Pseudocode for the recursive fatgraph folding problem.

Input : Finite number of fatgraphs $\gamma_1, \dots, \gamma_p$, sequence S , base-pair based energy model \mathcal{E}
Output : Best-scoring arc-annotation for S , in the class specified by the fatgraphs

```

1 foreach fatgraph  $\gamma_i$  do
2   Compute minimal expansion  $G_i$  of fatgraph  $\gamma_i$  ▶ Linear time; see Section 3.2
3   Find minimum width tree decomposition  $\mathcal{T}$  for  $G_i$  ▶ FPT in  $tw$  using classic exact tree dec. algorithm
4   Transform  $\mathcal{T}$  into a canonical form tree dec  $\mathcal{T}'$  ▶ Polynomial time; see Section 4.1
5   Compute skeleton of  $\mathcal{T}'$  ▶ Linear time; see Section 4.1
6   Derive corresponding DP scheme ▶ Linear time; see Section 4.2
7 end
8 Use union of DP schemes to find MFE arc-annotation of  $S$  ▶ XP in  $tw$   $O(n^{tw+1})$ ; See Section 5

```

133 **Output:** An algorithm that efficiently predicts an arc-annotation for S that minimizes the
 134 free energy according to \mathcal{E} , where the search space is exactly the recursive expansions of the
 135 input fatgraphs.

136 Our main result is stated Algorithm 1, which overall represents a $O(n^{tw+1})$ -algorithm for
 137 the fatgraph-folding problem, with tw the maximum treewidth of any of the input fatgraph.
 138 Its first steps represent, a fixed-parameter tractable (FPT) pre-processing of the input fat
 139 graphs yielding DP equations for folding (see Figure 1), which may be reused for any other
 140 input sequence.

141 ▶ **Theorem 3** (Main result). *Algorithm 1 is an XP algorithm for the recursive fatgraph*
 142 *folding problem, parameterized by the maximum treewidth of the input fatgraphs.*

143 3 Minimal representative expansion of a fatgraph

144 The pre-processing step of Algorithm 1 relies on computing *tree decompositions* of the input
 145 fatgraph(s), and its overall complexity depends on their *treewidth*. This section will define
 146 these notions, and present our main structural result, namely: expanding the helices of a
 147 fatgraph to length 5 (i.e. 5 nested base pairs) yields a graph which is representative in
 148 terms of treewidth of all possible expansions of the fatgraph. It is tree decompositions of
 149 these minimal representative expansions that will be reinterpreted as dynamic programming
 150 schemes, to complete our pre-processing.

151 3.1 Treewidth and tree decompositions

152 ▶ **Definition 4.** A *tree decomposition* $\mathcal{T} = (T, \{X_i\}_{i \in V(T)})$ of a graph $G = (V, E)$ is a tree
 153 of subsets of vertices of G , called *bags*, verifying the following conditions:

- 154 ■ $\forall u \in V \exists i \in V(T)$ such that $u \in X_i$. (vertex representativity)
- 155 ■ $\forall (u, v) \in E \exists i \in V(T)$ such that $\{u, v\} \subset X_i$. (edge representativity)
- 156 ■ $T_u = \{i \in V(T) \mid u \in X_i\}$ must be connected. (vertex subtree property)

157 The *width* of a tree decomposition is the size of its biggest bag minus one, i.e. $\min_{i \in V(T)} |X_i| -$
 158 1. The *treewidth* of a graph G is then the best possible width of a tree decomposition of G .
 159 Intuitively, the lower the treewidth, the closer G is to being a tree. Treewidth is NP-HARD
 160 to compute [3], but fixed-parameter tractable: there is a $O(f(w) \cdot n)$ algorithm [5] deciding
 161 whether $tw(G) \leq w$ given G . Many polynomial heuristics are also known to yield reasonable
 162 results [8], and optimized exact solvers have also been developed [42, 18].

163 Notoriously, a wide variety of hard computational problems can be solved efficiently when
 164 restricted to graphs of bounded treewidth [7, 11], including in bioinformatics [44, 41, 37].

165 Such is the case of LiCoRNA [37], for pseudoknotted structure-sequence alignment, of which
 166 the algorithm presented in this paper can be seen as a generalization.

167 We will rely in the remainder of this section on some well known-properties for treewidth,
 168 which we recall here. First, taking any *minor* of G [24], i.e. performing any sequence or edge
 169 contractions, edge deletions and vertex deletions on G can only lower the treewidth. Second,
 170 degree-2 vertices can be contracted into their neighbors without changing the treewidth, as
 171 quickly stated below (proof in appendix). This implies in particular that any bulge in a helix
 172 of an RNA structure graph is inconsequential with respect to treewidth.

173 ► **Proposition 5.** *If u is a degree-2 vertex of G with neighbors $\{v, w\}$, and $G_{v \leftarrow u}$ is the graph
 174 obtained by contracting u into v in G then $tw(G) = tw(G_{v \leftarrow u})$*

175 Then, we import from [6] an inequality valid for any *separator* of G . A *separator* is a
 176 subset S of vertices of G such that $G \setminus S$ is composed of at least 2 connected components,
 177 which we write $\mathcal{C}_G(S)$. We then have:

178 ► **Proposition 6.** *If S is a separator of G , then $tw(G) \leq \max_{C \in \mathcal{C}_G(S)} tw(G[C \cup clique(S)])$
 179 with $G[C \cup clique(S)]$ the subgraph of G induced by $C \cup S$ augmented by edges making S a
 180 clique. In case of equality, we say that S is safe.*

181 **Proof.** Consider, for each $C \in \mathcal{C}_G(S)$, a tree decomposition \mathcal{T}_C of $G[C \cup clique(S)]$. Since
 182 these graphs contain S as a clique, each \mathcal{T}_C must have a bag X_C containing S entirely.
 183 Consider now the following tree decomposition for G , make a bag out of S , and connect
 184 X_C for each C to it. The resulting tree decomposition is valid for G , and its width is the
 185 left-hand-side of the inequality. ◀

186 Let us finish by noting that, in a tree decomposition, any intersection $S = X \cap Y$ of two
 187 adjacent bags is always a separator of G . To write down the proofs of the following section
 188 in a smoother fashion, we add the following two properties, whose proofs are delayed to the
 189 appendix:

190 ► **Proposition 7.** *A tree decomposition can always be locally modified such that, for any two
 191 adjacent bags X and Y and $S = X \cap Y$:*

- 192 ■ $|S| \leq tw(G)$
- 193 ■ S is minimal with respect to inclusion, i.e. removing any vertex from S makes it lose its
 194 separating properties.

195 3.2 Helices of length 5 are sufficient to obtain generalizable tree 196 decompositions

197 Given an RNA graph (with one vertex per nucleotide and one edge per base pair and backbone
 198 link, see Figure 3(a)), we call *perfect helix* a set of directly nested base pairs, resulting in
 199 the subgraph depicted on Figure 3(b). We call the number of nested base pairs its *length*,
 200 and denote it with l . With a slight abuse of language, we call such a subgraph a *helix*, even
 201 for general graphs. Our main structural result is to show that the treewidth of a graph G
 202 does not increase when extending a helix past a length of 5. Its proof relies on the following
 203 inequality, involving the graphs G_{\boxtimes} and G_{\boxminus} , obtained from G by replacing a helix H with
 204 either \boxtimes or \boxminus , (see Figure 3(c)).

► **Lemma 8.** *Given a graph G and a helix H of length $l \geq 3$ in G , we have:*

$$tw(G_{\boxtimes}) - 1 \leq tw(G_{\boxminus}) \leq tw(G) \leq \max(4, tw(G_{\boxtimes}))$$

205 **Proof.** To start with, by noticing that the 4 extremities of the helix form a separator S
 206 between the inside and the outside of it, we get by Proposition 6 that $tw(G) \leq \max(H \cup$
 207 $clique(S), G_{\boxtimes})$. The graph $H \cup clique(S)$ does not depend on G , and consists of a helix with
 208 the 4 extremities forming a clique. With $l \geq 2$, it turns out that this graph has treewidth 4,
 209 see Appendix A, hence the inequality.

210 Next, we notice that G_{\boxminus} is a minor of G when $l \geq 3$. This can be seen by contracting
 211 the helix according to the pattern outlined on Figure 3(d) by the green areas (each green
 212 area is contracted to the extremity it contains). Therefore, $tw(G_{\boxminus}) \leq tw(G)$.

213 Finally, let us note that G_{\boxtimes} and G_{\boxminus} only differ by 1 edge, and removing a single
 214 edge from a graph can only decrease its treewidth by at most 1. Indeed, suppose that
 215 $tw(G_{\boxminus}) < tw(G_{\boxtimes}) - 1$, and consider an optimal tree decomposition \mathcal{T} for G_{\boxminus} . Let us denote
 216 by u and v the two extremities of the helix not connected in G_{\boxminus} . If the subtrees of bags
 217 containing respectively u and v do not intersect, then one can just add v to all bags of the
 218 tree decomposition, to represent the edge (u, v) while increasing the width by ≤ 1 . Therefore
 219 $tw(G_{\boxtimes}) - 1 \leq tw(G_{\boxminus})$ and the inequality is complete. ◀

220 Through the introduction of G_{\boxtimes} and G_{\boxminus} as the two possible graphs to which G is equivalent
 221 in terms of treewidth, Lemma 8 already contains the essence of our main structural result,
 222 Theorem 9. It will be the basis for generalizing tree decompositions of minimal expansions
 223 of a fatgraph to arbitrary helix lengths. Its proof is delayed to Appendix E.

224 ▶ **Theorem 9.** *If H is a helix in G of length $l \geq 5$, then extending the helix to have length*
 225 *$l + 1$ no longer increases the treewidth.*

226 Since bulges in a helix only consist of vertices of degree exactly 2, combining Proposition 5
 227 with Theorem 9 implies that the treewidth of any expansion of a given fatgraph is always
 228 smaller or equal than the treewidth of a minimal expansion where all bands are helices of
 229 length exactly 5, formally defined as:

230 ▶ **Definition 10** (Minimal representative expansion of a fatgraph). *Given a fatgraph γ , its*
 231 *minimal representative expansion consists of:*

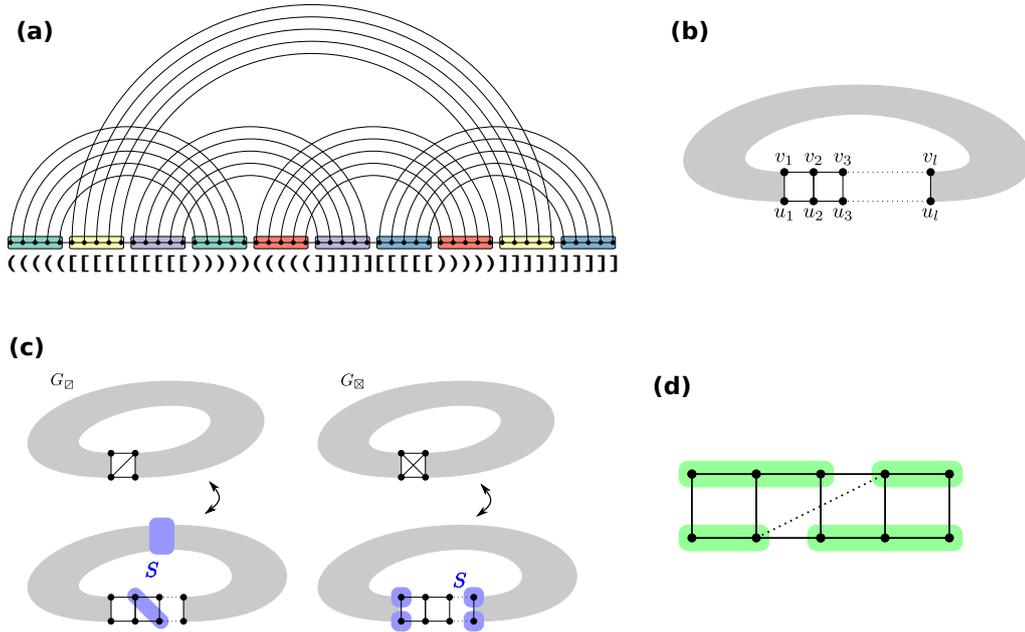
- 232 ■ *A perfect helix of length 5 for each band.*
- 233 ■ *No gap between the extremities of two helices*

234 Such a minimal representative expansion is illustrated in Figure 5(a). For visual clarity,
 235 gaps have been kept between consecutive helices, but one can see that the corresponding
 236 extremities have the same labels. Given a fatgraph, this RNA structure graph contains
 237 all necessary information for formulating DP equations decomposing all RNA structures
 238 compatible with the fatgraph.

239 Interestingly, the two graphs G_{\boxtimes} and G_{\boxminus} that emerge in the proofs as the two graphs G
 240 could be equivalent in terms of treewidth, as well as the separators they are associated to
 241 (see Figure 3 (c)) are reminiscent of two typical decomposition strategy used into dynamic
 242 programming for RNA folding. They suggest, for each helix in a graph, two possible
 243 “canonical representations” in terms of tree decomposition, which will be elaborated on in
 244 the next section.

245 4 Tree decompositions of fatgraph expansions as RNA DP algorithms

246 Starting with a tree decomposition for a minimal representative expansion of a given fatgraph,
 247 we first describe in this section how to put it in a *canonical form*, with each helix represented
 248 either in one of two different ways, respectively related to G_{\boxminus} and G_{\boxtimes} . The resulting tree



■ **Figure 3** (a) minimal expansion of a fatgraph, with every helix of length 5, and no unpaired base. The associated graph consists of one vertex per base, and one edge per base pair and backbone link. (b) A helix of length l in an RNA graph, as per the latter definition. (c) Given a helix in a graph G , the treewidth of G is either equal to $tw(G_{\square})$ or $tw(G_{\boxtimes})$. Each case is associated with a type of *separator* that can be used to extend the helix, or insert bulges, without changing the treewidth. (d) The dotted line represents a “hop-edge” which, if represented in a given tree decomposition of G , can be used to obtain G_{\square} as a minor of G , showing that the helix is in the “clique” case.

249 decomposition can be further compressed into a *skeleton*, where bags within individual helices
 250 are compressed into a single bag.

251 This tree can then be interpreted as a dynamic programming scheme, in which helices
 252 are generated by specializing dynamic programming subroutines. In a sense, the tree
 253 decomposition yields automatically a decomposition strategy usable for dynamic programming,
 254 of the kind that was hand-crafted in previous approaches [32, 14].

255 4.1 Canonical form for tree decompositions

256 We introduce an additional definition for the sake of presentation: Given an edge $e = (X, Y)$
 257 of a tree decomposition \mathcal{T} , we call the X – *side* of \mathcal{T} the connected component of $T \setminus e$
 258 containing X .

259 ► **Definition 11.** A tree decomposition of an expansion G of a fatgraph is in canonical form
 260 if, for each helix H of length l , either:

261 ■ **Clique case:** Helix H is represented by a root bag that contains all 4 extremities of H ,
 262 connected to a sub-tree-decomposition T_l recursively defined as

263
$$T_0^{\square} = \emptyset \quad T_l^{\square} = \{u_1, v_1, u_l, v_l\} \rightarrow \{u_1, v_1, u_l, v_{l-1}, v_l\} \rightarrow \{u_1, v_1, u_{l-1}, u_l, v_{l-1}\} \rightarrow T_{l-1}^{\square}.$$

265 ■ **Diagonal case:** Helix H is represented by a linear series of bags starting with $X_1 =$
 266 $S^* \cup \{u_1, v_1\}$, finishing with $X_{2l+2} = S^* \cup \{u_l, v_l\}$, and such that for $1 < k < l + 1$
 267 $X_{2k} = S^* \cup \{u_{2k-1}, v_{2k-1}, u_{2k}\}$ and $X_{2k+1} = S^* \cup \{v_{2k-1}, u_{2k}, v_{2k}\}$ for k odd.

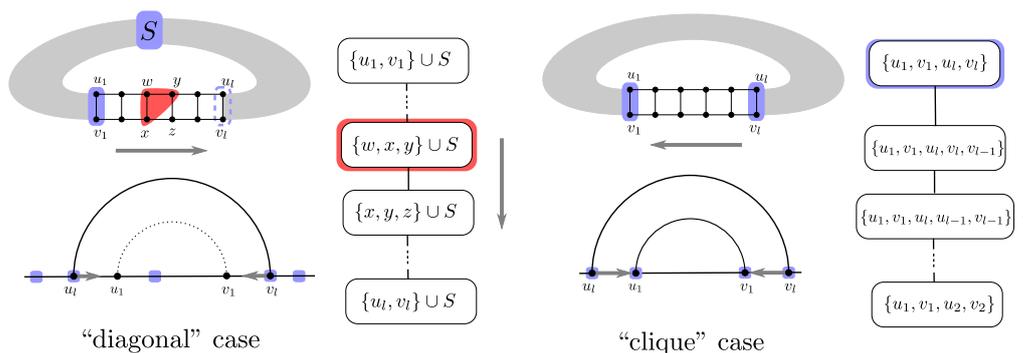


Figure 4 Illustration of the two types of canonical representations for the helices of a graph G

The definition above is illustrated on Figure 4. A canonical tree decomposition for a minimum expansion of a fatgraph is also presented on Figure 8. It was obtained through the processing routine that we describe in Algorithm 2 (see Appendix D), applicable to any (optimal or not) tree decomposition. It essentially follows the dichotomy of the proof of Theorem 9. We state its correctness and run-time below, but delay the proof to Appendix E.

► **Proposition 12.** *Given G and \mathcal{T} , Algorithm 2 outputs a canonical tree decomposition for G , having same width as \mathcal{T} , in time $O(N_H \cdot n^3)$, where N_H is the number of helices.*

Note that in a canonical tree decomposition, all vertices and edges internal to a helix of a graph are represented in the canonical sub-tree-decomposition associated to it. All bags outside of these canonical blocks only consist of extremities of helices, or other vertices outside of helices. Ignoring these internal parts, to focus on a more compact “skeleton” of canonical tree decompositions will be the first step towards automatically deriving dynamic programming equations.

► **Definition 13.** *The skeleton of a canonical tree decomposition for a graph G , is defined as follows:*

- All sub-tree-decompositions representing a helix in the “clique” case are replaced with a unique bag containing all extremities of the helix
- All sub-tree-decompositions representing a helix in the “diagonal” case are contracted to contain their first and last bags only, denoted as $S \cup \{u_1, v_1\}$ and $S \cup \{u_l, v_l\}$ in Definition 11.

Figure 5(b) gives an example of such a skeleton.

4.2 Automatic derivation of dynamic programming equations

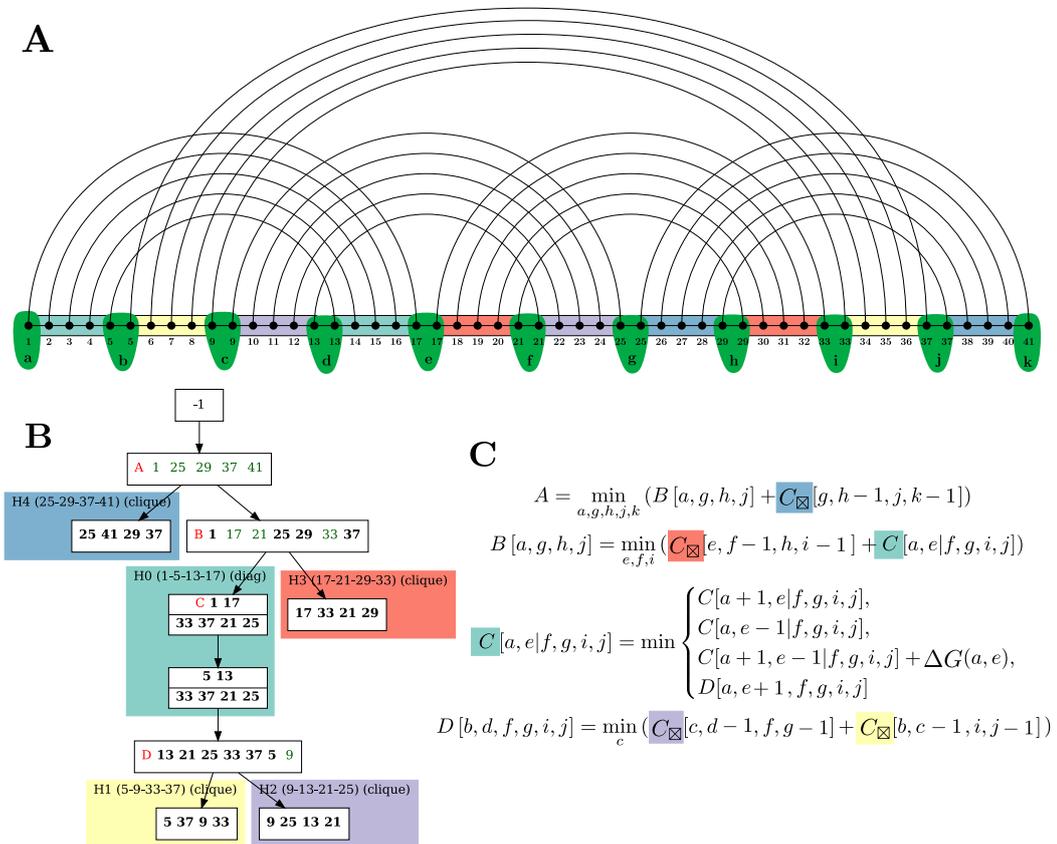
Given the skeleton of a representative minimal expansion of γ , we describe here how to formulate DP equations for the corresponding folding problem. As mentioned previously, we initially restrict our exposition to a base-pair based model, akin to the one optimized by the seminal Nussinov algorithm [28].

Essentially, we introduce helix DP tables for each helix, and transitional tables for non-helix bags. The variables indexing these tables are called *anchors*. These integer variables each represent a separation point between consecutive (half-)helices. Taken together, a full

298 set of anchors (a, b, c, \dots) partitions the sequence into a set of disjoint intervals $[a, b]$, $[b, c]$, \dots ,
 299 each associated with one *half-helix*, i.e. one of the subsequences that form a helix. Helix
 300 tables will account for the free-energy contributions of concrete base-pairs, while transitional
 301 tables will instantiate anchors in a way that remains consistent with previous assignments.

302 Indeed, owing to the tree decomposition, a skeleton is guaranteed to: i) feature each
 303 anchor in some bag; ii) represent each pair of consecutive anchors in at least one bag; iii)
 304 propagate anchor values, such that the anchor values within helix tables remain consistent.
 305 Due to this observation, non-helix bags can simply propagate previously-assigned anchors,
 306 possibly assigning values to novel anchors (if any and constrained to remain consistent with
 307 the sequential order) to explore all possible partitions of the input RNA sequence.

308 Helix tables will predict concrete set of base pairs and account for their associated
 309 free-energy. In order to both prevent the double pairing of certain sequence positions, and to
 310 avoid ambiguity, we require (and enforce in the DP rules) that an anchor x , separating the



■ **Figure 5** (a) Minimal representative length-5 expansion of the fat graph shown in Figure 1. Anchor variables are highlighted in green. We introduce one such variable per gap between helices. (b) Skeleton of the tree decomposition. White boxes represent transitional bags, introducing/propagating indices, while colored boxes represent helices in the fatgraph (H0 to H4) with associated indices in the input structure. Red letters indicate tables of the dynamic programming algorithm. Green indices are novel indices, absent from a bag’s predecessor. (c) DP equations derived from the compact skeleton, involving the anchor variable defined above, and following the rules described in Section 4.2.

consecutive halves of two helices H and H' , implies the pairing of position x to the other half of H' , and the pairing of some position $x' < x$ as part of H . In other words, a helix H delimited by anchors i, i', j' , and j must pair position i to some position $x \in]j', j]$, and j' to some position $y \in]i, i'[$, implicitly leaving both regions $]y, i'[$ and $]x, j[$ unpaired.

4.2.1 Helix table 1: “Clique” cases

In the skeleton, each bag representing a helix in the “clique” case is associated to the following tables, where $i, i' + 1, j'$, and $j + 1$ represent the values of the anchors delimiting the helix. The increments on i' and j are here to ensure the presence of gap of length ≥ 1 between two base pairs belonging to different helices. (see also Figure 5(c) for an example of how anchor values are passed to C_{\boxtimes} with a decrement of -1 for the same reason).

A first table C'_{\boxtimes} holds the minimal free-energy of a helix delimited by i, i', j' , and j , such that position i is paired to some $x \in]j', j]$ and j' to some position $y \in]i, i'[$. The idea is here to iteratively move the anchor from j to $j - 1$, implicitly leaving position j unpaired, until a base pair (i, j) is formed. Once a base pair is created, we transition to another table C_{\boxtimes} which optimizes over helices like C'_{\boxtimes} , but additionally allows position i to be left unpaired.

Those two tables can be filled owing to the following recurrences:

$$C'_{\boxtimes}[i, i', j', j] = \min \begin{cases} C'_{\boxtimes}[i, i', j', j - 1] & \text{if } j' < j \\ C_{\boxtimes}[i + 1, i', j', j - 1] + \Delta G_{i,j} & \text{if } (i < i') \wedge (j' < j) \\ \Delta G_{i,j} & \text{if } j = j' \\ +\infty & \text{if no such case apply} \end{cases}$$

and

$$C_{\boxtimes}[i, i', j', j] = \min \begin{cases} C'_{\boxtimes}[i, i', j', j - 1] & \text{if } j' < j \\ C_{\boxtimes}[i + 1, i', j', j] & \text{if } i < i' \\ C_{\boxtimes}[i + 1, i', j', j - 1] + \Delta G_{i,j} & \text{if } (i < i') \wedge (j' < j) \\ \Delta G_{i,j} & \text{if } j = j' \\ +\infty & \text{if no such case apply} \end{cases}$$

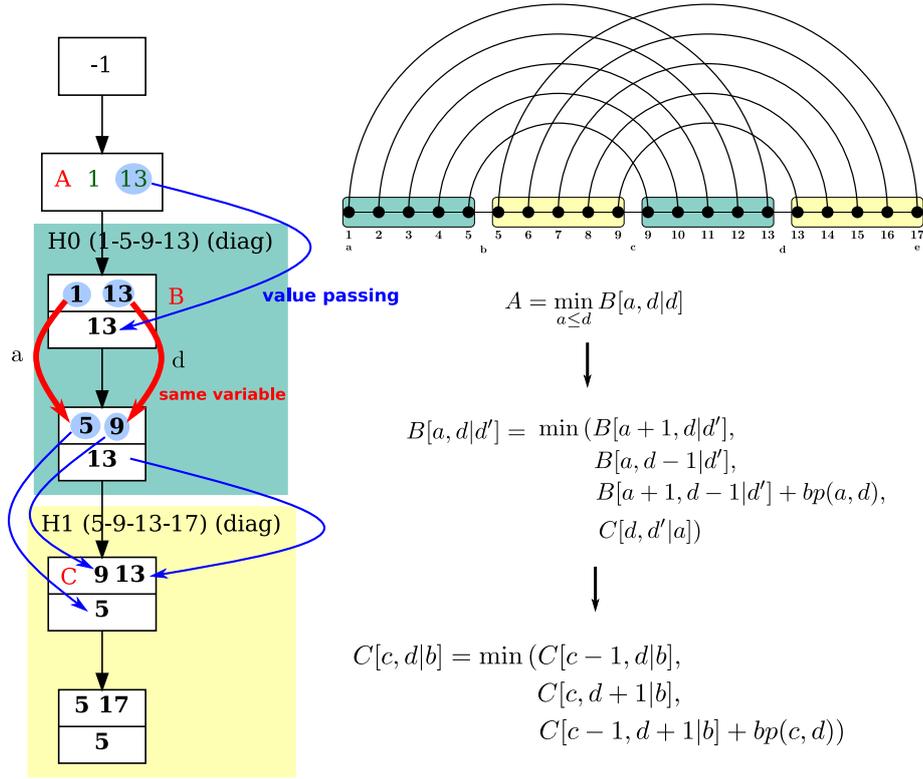
where $\Delta G_{i,j}$ denote the free-energy contribution of the base-pair (i, j) in the input RNA sequence.

4.2.2 Helix tables 2: “Diagonal” cases

In the skeleton bags representing the diagonal cases, we need to associate a different table to each helix. Indeed, each “diagonal” case associates, to a helix H , a set S of indices, dubbed the *constant anchors*, whose values remain unchanged during the construction of H .

We focus on the case where (i, j) represents the value of the outermost anchor pair (i.e. $[i, j]$ represents the full span of H), leaving to the reader the symmetric case starting from the innermost pair. Note that, in the skeleton, we kept two bags for a “diagonal case” helix. Yet they are associated to a single table, since the helix is created by incrementing two indices only, such that the initial pair of extremities “becomes” the other pair. We need this second bag to know how to map index values to the children tables $\{M_k\}_k$. This value mapping at the end of a diagonal case is illustrated on Figure 6.

Namely, let the cell $D_H[i, j | S]$ (resp. $D'_H[i, j | S]$) represent the minimum-free energy achieved by the set of helices in the subtree of H , when H is anchored at (i, j) without



■ **Figure 6** Derivation of DP equations from a skeleton, starting from the canonical tree decomposition of a length-5 expansion for a simple H -type fatgraph. On the left-hand-side, special emphasis is given to explaining how values are mapped at the end of a diagonal case. Extra tables C'_{\square} and D'_H , needed to ensure unambiguity of the DP scheme, are omitted for the sake of simplicity without adverse consequences to correctness.

constraints on i or j (resp. such that i is paired to some position $x \leq j'$). We have:

$$D'_H[i, j | S] = \min \begin{cases} D'_H[i, j-1 | S] & \text{if } j-1 > i \wedge \forall s \in S, j-1 \neq s \\ D_H[i+1, j-1 | S] + \Delta G_{i,j} & \text{if } \forall s \in S, (i+1 \neq s) \wedge (j-1 \neq s) \end{cases}$$

and

$$D_H[i, j | S] = \min \begin{cases} D_H[i+1, j | S] & \text{if } i+1 < j \wedge \forall s \in S, i+1 \neq s \\ D'_H[i, j-1 | S] & \text{if } j-1 > i \wedge \forall s \in S, j-1 \neq s \\ D_H[i+1, j-1 | S] + \Delta G_{i,j} & \text{if } \forall s \in S, (i+1 \neq s) \wedge (j-1 \neq s) \\ \sum_k M_k[I_k] & \text{with } I_k := (\{i, j+1\} \cup S) \cap A_k \end{cases}$$

339 where A_k denotes the anchors values needed for the k -th child of the diagonal bag.

340 4.2.3 Transitional tables: Non-helix bags

The general case consists of passing the values of relevant variables onward to the diagonal and clique tables, possibly assigning/propagating anchors that appear in the bag for the first

time. Let I_P be the anchors of the parent bag of M in the tree decomposition, we have:

$$M[I_P] = \min_{\substack{\text{Values for all} \\ \text{anchors in } I \setminus I_P}} \sum_k \begin{cases} M_k[I_k] & \text{if } k\text{-th child is transitional} \\ C'_{\boxtimes}[i, i' - 1, j', j - 1] & \text{if clique, anchored at } (i, i', j', j) \\ D'_{H_k}[i, j - 1 \mid S_k] & \text{if diagonal, anchored at } (i, j') \end{cases}$$

341 where I_k denotes the anchor values from I needed for the k -th child of the bag, and S
342 represents the constant anchors of the k -th child, assumed to be a diagonal.

343 5 Extensions

344 The DP scheme, as stated above, only supports conformations that consist of a single
345 pseudoknot configuration, indicated by a fatgraph. Moreover, it forces the first position
346 of the sequence to always form a base pair. Finally, it considers an energy model that is
347 fairly unrealistic in comparison with the current state of the art. In this section, we briefly
348 describe how to extend this fundamental construction in several directions in order to solve
349 the stated algorithm design problem (Def. 2) and consequently the associated folding problem
350 in complex energy models, and discuss the consequences on the complexity.

351 Multiple fatgraphs and integration within 2D folding scheme

352 Alternative fatgraphs can easily be considered, without added complexity, by simply adding
353 a disjunctive rule at the top level of the DP scheme, such as $\text{MFE}_{\text{PK}} := \min_{i=1}^p \text{root}_{\gamma_i}$
354 where root_{γ_i} is the top level of the DP scheme for fatgraph γ_i . The associated conformation
355 space then consists of the union of all pseudoknotted structures compatible with one of the
356 fatgraphs.

357 However, fatgraphs usually represent a structural module rather than a complete RNA
358 conformation. The classic DP scheme for 2D structure energy-minimization can thus be
359 supplemented by additional constructs, enabling the consideration of pseudoknots. Towards
360 that, one needs to access $\text{MFE}_{\text{PK}}(i, j)$, the MFE achieved over a region $[i, j]$ by a conformation
361 compatible with one of the input fat graphs. In other words, one needs an ability to prescribe
362 the span, say $[i, j]$, of the fatgraph occurrence, *i.e.* the values of the extremal anchors, while
363 initiating the dynamic programming.

To ensure this possibility, one simply needs to connect the first and last positions in the
minimal fatgraph completion. Indeed, since each arc of the input graph must be represented,
any tree decomposition for the completion will feature a bag B including both first and
last position (+ additional anchors $S := \{k_1, k_2, \dots\}$). Moreover, since a tree decomposition
is unordered, B can be arbitrarily used as the root, preceded by a root node restricted to
anchors (i, j) . This yields the following entry point for the DP of a fatgraph γ :

$$r_\gamma(i, j) := \min_{i < k_1 < k_2 < \dots < j} M_B[i, k_1, k_2, \dots, j]$$

364 which can be queried from within a classic DP scheme for the secondary structure.

365 Energy models

366 The extension to more realistic energy models is possible through functions evaluating
367 recursive non-crossing substructure; crossing configuration-specific score contributions; and
368 modifications of the algorithms that fill tables for the clique and diagonal cases. The former
369 enables scoring non-crossing substructure in the Turner model and doesn't require changes

370 beyond our discussion on recursive substructures and performing standard non-crossing free
 371 energy minimization. Handling multiple fatgraphs as described by disjunction at the top
 372 level enables specific scoring of different crossing configurations.

373 The latter case concerns the scoring of energy within helix expansions. Firstly, we observe
 374 that stacking energy between base pairs of the helix can be accounted for with minimal
 375 modification of the helix table recursions and therefore does not change the complexity. For
 376 this purpose, one introduces additional 'closed' states of the tables (corresponding to the
 377 matrix for closed subsequences in non-crossing free energy minimization). To explicitly score
 378 interior loops and bulges, the helix table recursions are extended by a case minimizing over
 379 the different loops. Naïvely, this would increase the complexity by a linear factor, which is
 380 avoided by bounding the loop size, as common in implemented folding algorithms, or without
 381 bounding the size following [25].

382 Recursive substructures

383 Recursive substructures consist of secondary structures/occurrences of fatgraphs that are
 384 inserted, both in between and within helices, usually through recursive calls to the (augmented)
 385 2D folding scheme.

386 To enable the insertion of substructures within an helix requires modifications to the helix
 387 clique/diagonal rules that are very similar to the ones enabling support for the Turner energy
 388 model. Assuming the presence of a base pair (i, j) , An insertion can indeed be performed
 389 by delimiting a region $[i, k]$ (resp. $[k, j]$) of arbitrary length, leading to an overall MFE of
 390 $\text{MFE}_{\text{SS}}(i, k) + \delta$, where δ is the free-energy contributed by the rest of the helix (possibly
 391 accounting for additional terms associated with multiloops).

392 To allow arbitrary sub-structures to be inserted in the gaps between consecutive helices,
 393 one can again modify the minimal helix expansion to distinguish the anchors a, b associated
 394 with consecutive helices (instead of merging them into a single anchor in our initial exposition).
 395 By connecting a and b , one ensures their simultaneous presence in a tagged bag B , whose
 396 DP recurrence is then augmented to include an energy contribution $\text{MFE}_{\text{SS}}(a + 1, b - 1)$.

397 Partition functions and ensemble applications

398 For ensemble applications of our DP schemes, such as computing the partition function [26]
 399 and statistical sampling of the Boltzmann ensemble [12], it is imperative for the DP scheme
 400 above to be complete and unambiguous [30]. Fortunately, both properties are already
 401 guaranteed by our DP schemes. Indeed, intuitively: the completeness is ensured by the
 402 exhaustive investigation of all possible anchor positions, i.e. all possible partitions; the
 403 unambiguity is guaranteed by the invariant that assigning a position x to a given anchor
 404 (within a transitional or diagonal bag), leads x to be paired within the (half-)helix immediately
 405 to its right. Choosing different values for x thus induces different innermost/outermost base
 406 pairs for the associated helix, leading to disjoint sets of structures.

407 From this property, we conclude that the partition function for a fatgraph (or several,
 408 possibly recursively and/or within a realistic energy model) can be obtained by simply
 409 replacing the $(\min, +, \Delta G)$ terms into $(\sum, \times, e^{\beta \Delta G})$, with $\beta = RT$ being the Boltzmann
 410 constant multiplied by some absolute temperature.

6 (Re-)Designing algorithms for specific pseudoknot classes

Since the algorithms in [32] have been described in terms of a finite number of fatgraphs (or there equivalently, shadows), one can directly apply our method to obtain an efficient algorithm that covers the same class as `gfold`, namely **1-structures** that are recursive expansions of the four fatgraphs of genus 1 corresponding to simple PK 'H' (`[]`), kissing hairpin 'K' (`[\]`), three-knot 'L' (`{[\]}`) and 'M' (`[{\}]`) (here, represented in *dot-bracket notation*, i.e. corresponding opening and closing brackets correspond to arcs). The maximum complexity of $O(n^6)$ of the four fatgraphs (see Table 1) implies that the automatically derived algorithm covers the class of 1-structures in $O(n^6)$ time—the same complexity as hand-crafted `gfold`. Note that [32] used declarative methods in their algorithm design only to the point of generating grammar rules, which without further optimization yield $O(n^{18})$ (after applying algebraic dynamic programming; ADP [36]). In contrast, our method obtains the optimal complexity in fully automatic fashion. Beyond this re-design of `gfold`, remarkably our method is equally prepared to automatically design a DP algorithm with optimized efficiency for **2-structures**, which are based on all genus 2 fatgraphs. This is remarkable, since the implementation of a practical algorithm has been considered infeasible [32] due to the large number of genus 2 shadows (namely, there are 3472 shadows/fatgraphs), whose grammar rules would have to be optimized by hand. In contrast, due to full automation, our method directly handles even the large number of fatgraphs of genus 2 and yields an efficient, complexity optimized, DP scheme.

Recall that we cover all other pseudoknot classes that are recursive expansions of a finite number of fatgraphs (in the same way as we cover the design of prediction algorithms for 1- and 2-structures). In this way, among the previously existing DP algorithms, we cover the class of **Dirks&Pierce** (D&P) [14], simply by specifying the H-type as single input fatgraph. Consequently, we automatically re-design the D&P algorithm in the same complexity of $O(n^5)$. Even more interestingly, we can design algorithms covering specific (sets of) crossing configurations. This results in an infinite class of efficient algorithms that have not been designed before. Again the complexity of such algorithms is dominated by the most complex fatgraph; where results for interesting ones are given in Table 1. Most remarkably, we design an algorithm optimizing over recursive expansions of kissing hairpins in $O(n^4)$, whereas CCJ [10, 21], which was specifically designed to cover kissing hairpins, requires $O(n^5)$.

A special case, which further showcases the flexibility, is the extension of existing classes by specific crossing configurations. For example, extending D&P by kissing hairpin covers a much larger class while staying in the same complexity. Extending 1-structures by 5-chain yields a new algorithm with a complexity below of 2-structures (namely only $O(n^7)$ instead of $O(n^8)$ [32]). The complexity of 5-chain is remarkably low, when considering that previously described algorithms covering this configuration take $O(n^8)$ (e.g. `gfold`'s generalization to 2-structures and a hypothetical blow-up of the Rivas and Eddy algorithm [38] to 6-dimensional instead of 4-dimensional DP matrix elements—both of which have never been implemented).

7 Conclusions

In this work, we provide an algorithm that takes a family of fatgraphs, i.e. pseudoknotted structures, and returns DP equations that efficiently predicts an arc annotations minimizing the free energy in optimal complexity. The DP equations are automatically generated based on an expansion of the fatgraph, designed to capture helices of arbitrary length. This very general framework recovers the complexity of prior, hand-crafted algorithms, and lays the foundation for a purely declarative approach to RNA folding with pseudoknots.

name	fatgraph	treewidth	complexity of folding
H-type	([])	4	$O(n^5)$
kissing hairpins	([][])	4	$O(n^4)$ (*)
“L” [32]	([{}])	5	$O(n^6)$
“M” [32]	([{}][{}])	5	$O(n^6)$
4-clique	([{}<])>	5	$O(n^6)$
5-clique	([{}<A])>a	5	$O(n^6)$
5-chain	([{}][{}])	6	$O(n^7)$

■ **Table 1** Table listing pseudoknot classes, corresponding treewidth and resulting complexity of the folding algorithm. In all cases except the one denoted by (*), the complexity of folding is equal to $O(n^{tw+1})$. For the kissing hairpins case, we are in the specific case where the most complex routine is the alignment of a “clique case” helix, which is done in $O(n^4)$ despite a treewidth of 4. These examples are detailed in the Appendix, Figure 9.

457 In addition to the extensions described in Section 5, this work suggests perspectives that
 458 will be explored in future work. Indeed, the choice of an optimal decomposition/DP scheme
 459 for the input fatgraph can be seen as the automated design of an optimal table strategy in
 460 the context of algebraic dynamic programming [31, 4, 36]. This would enable extensions to
 461 multiple context free grammars or tree grammars when describing the problem in the ADP
 462 framework.

463 Our automated design of pseudoknot folding algorithms could naturally be extended
 464 to RNA–RNA interactions, since the joint conformation of two interacting RNA sequences
 465 can be seen as a pseudoknot when concatenating the two structures [13]. More ambitiously,
 466 categories of pseudoknots inducing an infinite family of fatgraphs, *e.g.* as covered by the
 467 seminal Rivas & Eddy algorithm [38], could be captured by allowing the introduction of
 468 recursive gapped structures in prescribed part of the fatgraph. This could be addressed by
 469 adding cliques to the minimal completion graph would ensure the availability of the relevant
 470 anchors in some bags of the tree decomposition, allowing to score such, non-contiguous,
 471 recursive substructures.

472 Another avenue for future research includes a proof of optimality, in term of polynomial
 473 complexity, for the produced DP algorithms. Of course, it would be far too ambitious (and
 474 erroneous) to expect our DP schemes to be optimal within general computational models.
 475 However, it may be possible to prove optimality within a clearly-defined subset of standard
 476 implementations of a subset of DP schemes, *e.g.* by contradiction since the existence of a
 477 better algorithm would imply the existence of a tree decomposition having smaller width.

478 — References —

- 479 1 Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction
 480 with pseudoknots. *Discrete Applied Mathematics*, 104(1-3):45–62, 2000.
- 481 2 Can Alkan, Emre Karakoç, Joseph H. Nadeau, S. Cenk Sahinalp, and Kaizhong Zhang.
 482 RNA–RNA Interaction Prediction and Antisense RNA Target Search. *Journal of Computational
 483 Biology*, 13(2):267–282, 2006. doi:10.1089/cmb.2006.13.267.
- 484 3 Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings
 485 in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.
- 486 4 Sarah J Berkemer, Christian Höner zu Siederdisen, and Peter F Stadler. Algebraic dynamic
 487 programming on trees. *Algorithms*, 10(4):135, 2017.

- 488 5 Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth.
489 *SIAM Journal on computing*, 25(6):1305–1317, 1996.
- 490 6 Hans L Bodlaender and Arie MCA Koster. Safe separators for treewidth. *Discrete Mathematics*,
491 306(3):337–350, 2006.
- 492 7 Hans L Bodlaender and Arie MCA Koster. Combinatorial optimization on graphs of bounded
493 treewidth. *The Computer Journal*, 51(3):255–269, 2008.
- 494 8 Hans L Bodlaender and Arie MCA Koster. Treewidth computations i. upper bounds. *Inform-*
495 *ation and Computation*, 208(3):259–275, 2010.
- 496 9 Song Cao and Shi-Jie Chen. Predicting RNA pseudoknot folding thermodynamics. *Nucleic*
497 *Acids Research*, 34(9):2634–2652, 01 2006. doi:10.1093/nar/gkl346.
- 498 10 Ho-Lin Chen, Anne Condon, and Hosna Jabbari. An $O(n^5)$ algorithm for MFE prediction of
499 kissing hairpins and 4-chains in nucleic acids. *Journal of Computational Biology*, 16(6):803–815,
500 2009.
- 501 11 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin
502 Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer,
503 2015.
- 504 12 Ye Ding and Charles E. Lawrence. A statistical sampling algorithm for RNA secondary structure
505 prediction. *Nucleic Acids Research*, 31(24):7280–7301, 12 2003. doi:10.1093/nar/gkg938.
- 506 13 Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce.
507 Thermodynamic analysis of interacting nucleic acid strands. *SIAM review*, 49(1):65–88, 2007.
- 508 14 Robert M Dirks and Niles A Pierce. A partition function algorithm for nucleic acid secondary
509 structure including pseudoknots. *Journal of computational chemistry*, 24(13):1664–1677, 2003.
- 510 15 Chuong B Do, Daniel A Woods, and Serafim Batzoglou. CONTRAfold: RNA secondary
511 structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–e98, 2006.
- 512 16 Mark E. Fornace, Nicholas J. Porubsky, and Niles A. Pierce. A Unified Dynamic Programming
513 Framework for the Analysis of Interacting Nucleic Acid Strands: Enhanced Models, Scalability,
514 and Speed. *ACS Synthetic Biology*, 9(10):2665–2678, 2020. PMID: 32910644. doi:10.1021/
515 acssynbio.9b00523.
- 516 17 Robert Giegerich, Björn Voß, and Marc Rehmsmeier. Abstract shapes of rna. *Nucleic acids*
517 *research*, 32(16):4843–4851, 2004.
- 518 18 Vibhav Gogate and Rina Dechter. A complete anytime algorithm for treewidth. *arXiv preprint*
519 *arXiv:1207.4109*, 2012.
- 520 19 Fenix Huang, Christian Reidys, and Reza Rezazadegan. Fatgraph models of RNA structure.
521 *Computational and Mathematical Biophysics*, 5(1):1–20, 2017.
- 522 20 Hosna Jabbari and Anne Condon. A fast and robust iterative algorithm for prediction of RNA
523 pseudoknotted secondary structures. *BMC bioinformatics*, 15(1):1–17, 2014.
- 524 21 Hosna Jabbari, Ian Wark, Carlo Montemagno, and Sebastian Will. Knotty: efficient and
525 accurate prediction of complex RNA pseudoknot structures. *Bioinformatics*, 34(22):3849–3856,
526 2018.
- 527 22 Martin Loebel and Iain Moffatt. The chromatic polynomial of fatgraphs and its categorification.
528 *Advances in Mathematics*, 217(4):1558–1587, 2008.
- 529 23 R Lorenz, SH Bernhart, C Höner Zu Siederdisen, H Tafer, C Flamm, PF Stadler, and
530 IL Hofacker. ViennaRNA Package 2.0. vol. 6. *Algorithms Mol. Biol*, page 26, 2011.
- 531 24 László Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–
532 86, 2006.
- 533 25 R. B. Lyngsø, M. Zuker, and C. N. Pedersen. Fast evaluation of internal loops in RNA
534 secondary structure prediction. *Bioinformatics (Oxford, England)*, 15(6):440–445, June 1999.
535 doi:10.1093/bioinformatics/15.6.440.
- 536 26 J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for rna
537 secondary structure. *Biopolymers*, 29(6-7):1105–1119, 1990. doi:https://doi.org/10.1002/
538 bip.360290621.

- 539 **27** Mathias Möhl, Sebastian Will, and Rolf Backofen. Lifting prediction to alignment of RNA
540 pseudoknots. *Journal of Computational Biology*, 17(3):429–442, 2010.
- 541 **28** Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of
542 single-stranded rna. *Proceedings of the National Academy of Sciences*, 77(11):6309–6313, 1980.
- 543 **29** Robert Clark Penner, Michael Knudsen, Carsten Wiuf, and Jørgen Ellegaard Andersen.
544 Fatgraph models of proteins. *Communications on Pure and Applied Mathematics*, 63(10):1249–
545 1297, 2010.
- 546 **30** Yann Ponty and Cédric Saule. A combinatorial framework for designing (pseudoknotted)
547 RNA algorithms. In Teresa M. Przytycka and Marie-France Sagot, editors, *Algorithms in*
548 *Bioinformatics*, pages 250–269, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- 549 **31** Michela Quadrini, Luca Tesei, and Emanuela Merelli. An algebraic language for RNA
550 pseudoknots comparison. *BMC bioinformatics*, 20(4):1–18, 2019.
- 551 **32** Christian M Reidys, Fenix WD Huang, Jørgen E Andersen, Robert C Penner, Peter F
552 Stadler, and Markus E Nebel. Topology and prediction of RNA pseudoknots. *Bioinformatics*,
553 27(8):1076–1085, 2011.
- 554 **33** Christian M Reidys and Rita R Wang. Shapes of RNA pseudoknot structures. *Journal of*
555 *Computational Biology*, 17(11):1575–1590, 2010.
- 556 **34** Jihong Ren, Baharak Rastegari, Anne Condon, and Holger H Hoos. HotKnots: heuristic
557 prediction of RNA secondary structures including pseudoknots. *Rna*, 11(10):1494–1504, 2005.
- 558 **35** Jessica S Reuter and David H Mathews. RNAstructure: software for rna secondary structure
559 prediction and analysis. *BMC bioinformatics*, 11(1):1–9, 2010.
- 560 **36** Maik Riechert, Christian Höner zu Siederdissen, and Peter F. Stadler. Algebraic dynamic
561 programming for multiple context-free grammars. *Theoretical Computer Science*, 639:91–109,
562 August 2016. doi:10.1016/j.tcs.2016.05.032.
- 563 **37** Philippe Rinaudo, Yann Ponty, Dominique Barth, and Alain Denise. Tree decomposition and
564 parameterized algorithms for RNA structure-sequence alignment including tertiary interactions
565 and pseudoknots. In *International Workshop on Algorithms in Bioinformatics*, pages 149–164.
566 Springer, 2012.
- 567 **38** Elena Rivas and Sean R Eddy. A dynamic programming algorithm for RNA structure prediction
568 including pseudoknots. *Journal of molecular biology*, 285(5):2053–2068, 1999.
- 569 **39** Kengo Sato, Manato Akiyama, and Yasubumi Sakakibara. RNA secondary structure prediction
570 using deep learning with thermodynamic integration. *Nature communications*, 12(1):1–9, 2021.
- 571 **40** Kengo Sato, Yuki Kato, Michiaki Hamada, Tatsuya Akutsu, and Kiyoshi Asai. IPknot:
572 fast and accurate prediction of RNA secondary structures with pseudoknots using integer
573 programming. *Bioinformatics*, 27(13):i85–i93, 2011.
- 574 **41** Céline Scornavacca and Mathias Weller. Treewidth-based algorithms for the small parsimony
575 problem on networks. In *WABI*, volume 201 of *LIPICs*, pages 6:1–6:21. Schloss Dagstuhl -
576 Leibniz-Zentrum für Informatik, 2021.
- 577 **42** Hisao Tamaki. Positive-instance driven dynamic programming for treewidth. *Journal of*
578 *Combinatorial Optimization*, 37(4):1283–1311, 2019.
- 579 **43** Edwin Ten Dam, Kees Pleij, and David Draper. Structural and functional aspects of RNA
580 pseudoknots. *Biochemistry*, 31(47):11665–11676, 1992.
- 581 **44** Hua-Ting Yao, Jérôme Waldspühl, Yann Ponty, and Sebastian Will. Taming Disruptive Base
582 Pairs to Reconcile Positive and Negative Structural Design of RNA. In *RECOMB 2021-25th*
583 *international conference on research in computational molecular biology*, 2021.
- 584 **45** Shay Zakov, Yoav Goldberg, Michael Elhadad, and Michal Ziv-Ukelson. Rich parameterization
585 improves RNA structure prediction. *Journal of Computational Biology*, 18(11):1525–1542,
586 2011.
- 587 **46** Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic*
588 *acids research*, 31(13):3406–3415, 2003.

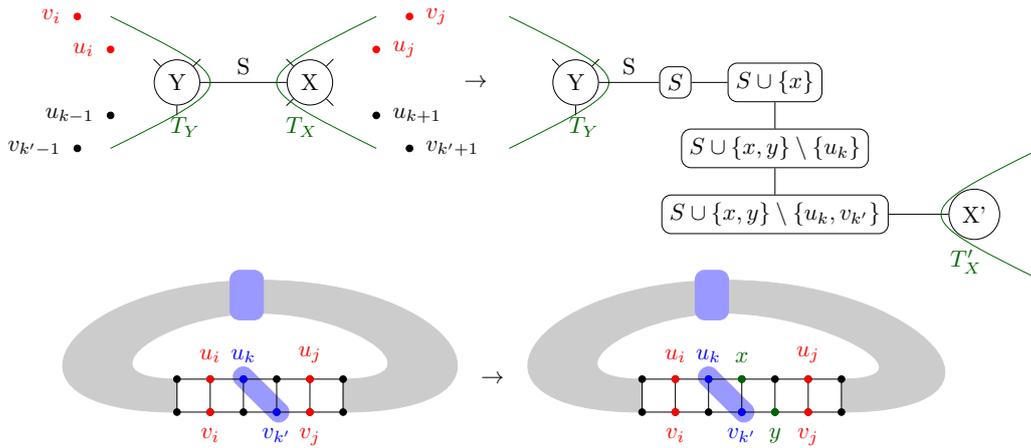


Figure 7 Representation of the local rewriting of a tree decomposition next to a separator S separating to base pairs (u_i, v_i) and (u_j, v_j) , in order to extend a helix by one unit, through the introduction of new vertices x and y .

A Width of a helix closed by a clique

589

590 Let us denote by H_l^* the graph corresponding to a helix of length l , with the extremities
 591 connected as a clique. This graph appears when considering the possible safety (see
 592 Proposition 6) of the extremities as a separator of the graph. We show the following result:

593 **► Lemma 14.** For $l = 2$, $tw(H_l^*) = 3$, while for $l \geq 3$, $tw(H_l^*) = 4$.

594 **Proof.** For $l = 2$, H_l^* is simply the clique on 4 vertices, and which has a width of 3. For $l \geq 3$,
 595 a clique on 5 vertices can be obtained as a minor by contracting the internal part of the helix
 596 to one vertex, which ends up being connected to all 4 extremities, which already form a clique.
 597 Therefore, $tw(H_l^*) \geq 4$. To obtain the equality, we recursively build a tree decomposition
 598 of width ≤ 4 , starting with $l = 2$ which we already described. Given a tree decomposition
 599 of width ≤ 4 for H_l^* , there has to be a bag X containing all 4 extremities $\{u_1, v_1, u_l, v_l\}$
 600 (see Figure 3(b)). We introduce two new bags: $X' = \{u_1, v_1, u_l, v_l, v_{l+1}\}$ introducing a new
 601 vertex v_{l+1} , and $X'' = \{u_1, v_1, u_l, v_{l+1}, u_{l+1}\}$ introducing u_{l+1} . We connect X' to X and X''
 602 to X' . By doing so, we respect the subtree connectivity property for all involved vertices,
 603 and build a tree decomposition capable of representing H_{l+1}^* . ◀

B Helix extension close to a separator

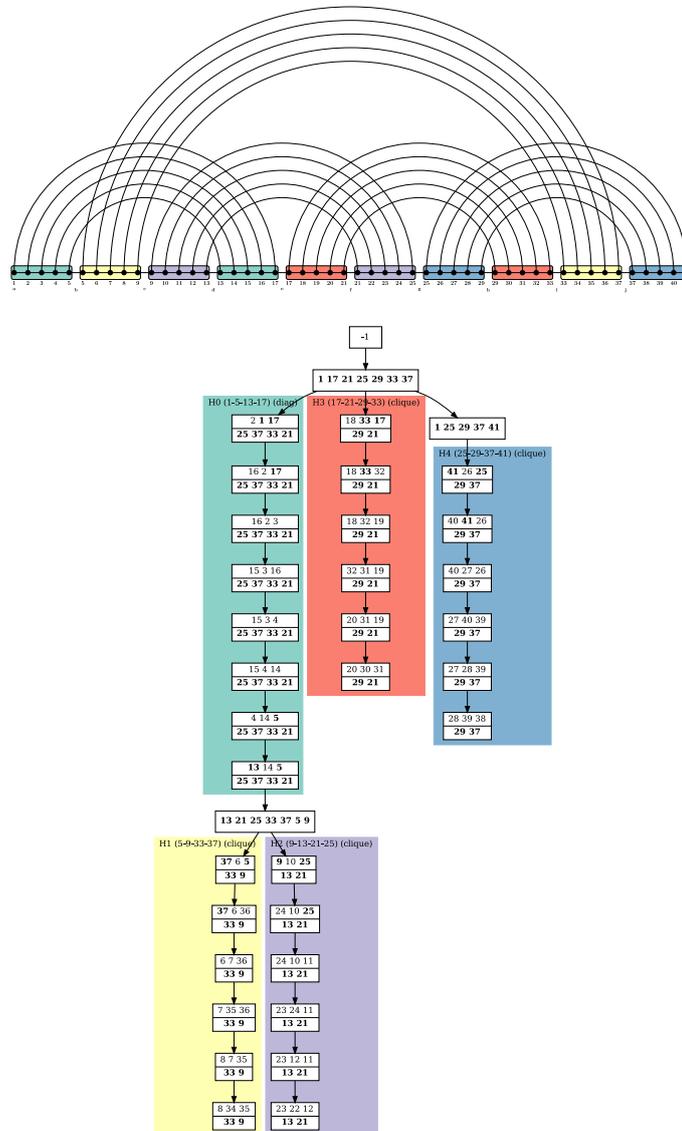
604

605 Figure 7 shows how, once we have found a separator, associated to an edge of the tree
 606 decomposition, separating $\{u_i, v_i\}$ from $\{u_j, v_j\}$ with $i < j$, we can insert new vertices in the
 607 helix, extending it while preserving the treewidth. This is used in the proof of Theorem 9, in
 608 what corresponds in Section 4 to the “diagonal” case.

C Detailed examples

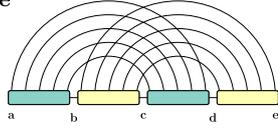
609

610 Figure 8 shows a canonical tree decomposition for the minimal length-5 expansion, shown in
 611 the upper half of the figure, for the fatgraph showed in Figure 1. This tree decomposition
 612 is optimal, and was computed with [42], a solver that empirically works quite fast on RNA
 613 graphs.



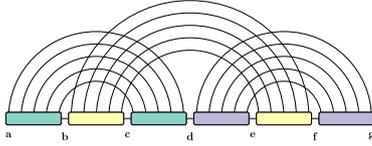
■ **Figure 8** Canonical tree decomposition of the fatgraph given in Figure 1. White boxes represent the bags of the tree decomposition. Number in the bags correspond to the indices of the helices in the fatgraph where number on the bottom are kept while traversing the branch of the decomposition tree. Colored frames indicate the distinct helices (H0 to H4) of the structure.

H-type



$$A = \min_{a,b,c,d,e} (C_{\boxtimes} [b, c-1, d, e-1] + C_{\boxtimes} [a, b-1, c, d-1])$$

kissing hairpins



$$A = \min_{a,d,g} B [a, d, d, g]$$

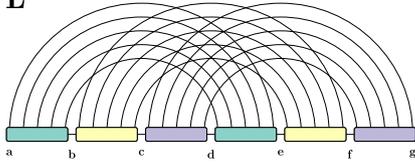
$$B [a, d, d', g] = \min \begin{cases} B' [a, d-1 | d', g], & \text{if } d-1 \notin \{a, d', g\} \\ B [a+1, d-1 | d', g] + \Delta G(a, d) & \text{if } \{a+1, d-1\} \cap \{d', g\} = \emptyset \end{cases}$$

$$B' [a, d | d', g] = \min \begin{cases} B [a+1, d | d', g], & \text{if } a+1 \notin \{d, d', g\} \\ B' [a, d-1 | d', g], & \text{if } d-1 \notin \{a, d', g\} \\ B [a+1, d-1 | d', g] + \Delta G(a, d) & \text{if } \{a+1, d-1\} \cap \{d', g\} = \emptyset, \end{cases}$$

$$C' [d, g | b, c] = \min \begin{cases} C' [d, g-1 | b, c], & \text{if } g-1 \notin \{d, b, c\} \\ C [d+1, g-1 | b, c] + \Delta G(d, g) & \text{if } \{d+1, g-1\} \cap \{b, c\} = \emptyset \end{cases}$$

$$C [d, g | b, c] = \min \begin{cases} C [d+1, g | b, c], & \text{if } d+1 \notin \{g, b, c\} \\ C' [d, g-1 | b, c], & \text{if } g-1 \notin \{d, b, c\} \\ C [d+1, g-1 | b, c] + \Delta G(d, g) & \text{if } \{d+1, g-1\} \cap \{b, c\} = \emptyset, \\ C_{\boxtimes} [b, c-1, d, g+1-1] & \end{cases}$$

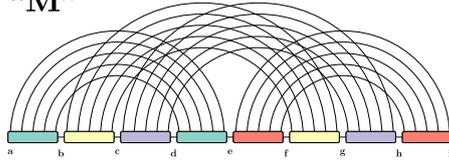
“L”



$$A = \min_{a,c,d,f,g} (B [a, c, d, f] + C_{\boxtimes} [c, d-1, f, g-1])$$

$$B [a, c, d, f] = \min_{b,e} (C_{\boxtimes} [b, c-1, e, f-1] + C_{\boxtimes} [a, b-1, d, e-1])$$

“M”

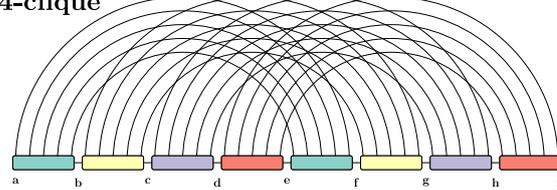


$$A = \min_{a,e,f,h,i} (B [a, e, f, h] + C_{\boxtimes} [e, f-1, h, i-1])$$

$$B [a, e, f, h] = \min_{b,d} (C_{\boxtimes} [a, b-1, d, e-1] + C [b, d, f, h])$$

$$C [b, d, f, h] = \min_{c,g} (C_{\boxtimes} [c, d-1, g, h-1] + C_{\boxtimes} [b, c-1, f, g-1])$$

4-clique

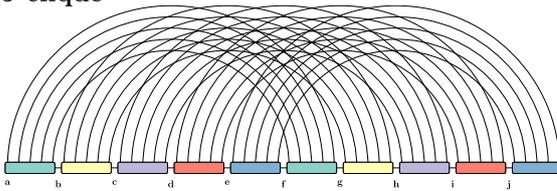


$$A = \min_{a,d,e,h,i} (B [a, d, e, h] + C_{\boxtimes} [d, e-1, h, i-1])$$

$$B [a, d, e, h] = \min_{c,g} (C [a, c, e, g] + C_{\boxtimes} [c, d-1, g, h-1])$$

$$C [a, c, e, g] = \min_{b,f} (C_{\boxtimes} [b, c-1, f, g-1] + C_{\boxtimes} [a, b-1, e, f-1])$$

5-clique



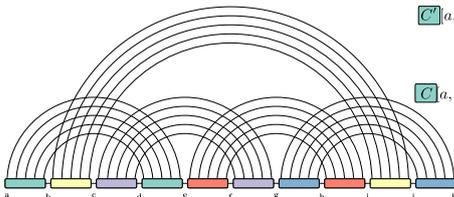
$$A = \min_{a,e,f,j,k} (B [a, e, f, j] + C_{\boxtimes} [e, f-1, j, k-1])$$

$$B [a, e, f, j] = \min_{d,i} (C [a, d, f, i] + C_{\boxtimes} [d, e-1, i, j-1])$$

$$C [a, d, f, i] = \min_{b,g} (D [b, d, g, i] + C_{\boxtimes} [a, b-1, f, g-1])$$

$$D [b, d, g, i] = \min_{c,h} (C_{\boxtimes} [c, d-1, h, i-1] + C_{\boxtimes} [b, c-1, g, h-1])$$

5-cycle



$$A = \min_{a,g,h,j,k} (B [a, g, h, j] + C_{\boxtimes} [g, h-1, j, k-1])$$

$$B [a, g, h, j] = \min_{e,f,i} (C_{\boxtimes} [e, f-1, h, i-1] + C [a, e, f, g, i, j])$$

$$C' [a, e, f, g, i, j] = \min \begin{cases} C' [a, e-1 | f, g, i, j], & \text{if } e-1 \notin \{a, f, g, i, j\} \\ C [a+1, e-1 | f, g, i, j] + \Delta G(a, e) & \text{if } \{a+1, e-1\} \cap \{f, g, i, j\} = \emptyset \end{cases}$$

$$C [a, e, f, g, i, j] = \min \begin{cases} C [a+1, e | f, g, i, j], & \text{if } a+1 \notin \{e, f, g, i, j\} \\ C' [a, e-1 | f, g, i, j], & \text{if } e-1 \notin \{a, f, g, i, j\} \\ C [a+1, e-1 | f, g, i, j] + \Delta G(a, e) & \text{if } \{a+1, e-1\} \cap \{f, g, i, j\} = \emptyset, \\ D' [a, e+1, f, g, i, j] & \end{cases}$$

$$D [b, d, f, g, i, j] = \min_c (C_{\boxtimes} [c, d-1, f, g-1] + C_{\boxtimes} [b, c-1, i, j-1])$$

Figure 9 Minimal representative expansions and final equations for the examples of Table 1. The equations have been automatically generated given optimal tree decompositions computed by [42]

614 **D** Transforming a tree decomposition into its canonical form

615 Algorithm 2 describes how to obtain a canonical tree decomposition for an RNA structure
 616 graph, given any valid tree decomposition as input. Interestingly, it can use a sub-optimal
 617 tree decomposition obtained from a polynomial heuristic [7] instead of an exponential solver
 618 (although [42] is empirically quite efficient on RNA structure graphs).

619 The run-time and correctness of Algorithm 2 are stated in Proposition 12.

ALGORITHM 2 Algorithm for re-writing a tree decomposition into a canonical one in which every helix of the input graph is represented in a canonical way

Input : A (not necessarily optimal) tree decomposition \mathcal{T} of a minimal expansion of a fatgraph γ .

Output : A tree decomposition of G in canonical form

```

1 if  $width(\mathcal{T}) \leq 3$  then
2   foreach helix  $H$  in fatgraph  $\gamma$  do
3     if  $\exists$  hop-edge represented in  $\mathcal{T}$  then
4       use hop-edge to obtain a tree dec. for  $G_{\boxtimes}$  //► (see Fig. 3(d))
5
6       find a bag  $X = \{u_1, v_1, u_l, v_l\}$  as  $w(T) \leq 3$ 
7       replace  $X$  with a “diagonal” canonical representation with  $S = \emptyset$ .
8     else
9       find an edge  $(X, Y)$  of  $\mathcal{T}$  s.t  $X \cap Y$  separates  $u_1, v_1$  on the X-side from  $u_l, v_l$  on
10      the Y-side
11       $\forall i$ , replace  $u_i$  with  $u_1$  and  $v_i$  with  $v_1$  in all bags of the X-side of  $\mathcal{T}$ 
12       $\forall j$ , replace  $u_j$  with  $u_l$  and  $v_j$  with  $v_l$  in all bags of the Y-side of  $\mathcal{T}$ 
13      Insert between  $X$  and  $Y$  the “diagonal” canonical representation for  $H$ , with
14      constant part  $S = (X \cap Y) \setminus \{u_k, v_k\}_{i \leq k \leq j}$ 
15     end
16   end
17 else
18   for helix  $H$  in  $\gamma$  do
19     if  $\exists$  a hop-edge represented in  $\mathcal{T}$  then
20       Use the hop-edge to obtain a tree decomposition for  $G_{\boxtimes}$ 
21       find a bag containing all extremities and connect  $T_l^{\boxtimes}$  to it
22     else
23       find an edge  $(X, Y)$  of  $\mathcal{T}$  separating  $\{u_1, v_1\}$  and  $\{u_l, v_l\}$ 
24        $\forall i$  replace  $u_i$  with  $u_1$  and  $v_i$  with  $v_1$  on the X-side of  $\mathcal{T}$ 
25        $\forall i$  replace  $u_i$  with  $u_l$  and  $v_i$  with  $v_l$  on the Y-side of  $\mathcal{T}$ 
26       Insert between  $X$  and  $Y$  the “diagonal” canonical representation for  $H$ , with
27       constant part  $S = (X \cap Y) \setminus \{u_k, v_k\}_{1 \leq k \leq l}$ 
28     end
29   end
30 end

```

620 **E** Delayed proofs

621 **Proof of Proposition 12.** Concerning the run-time, enumerating all pairs $1 \leq i < j \leq l$ is
 622 quadratic in the length of the helix under consideration, which is $O(n)$ in a general graph,
 623 while testing a given edge for separation of u_i, v_i and u_j, v_j takes $O(n)$ (through breadth-first
 624 search) for each of the $O(n)$ edges of the tree decomposition. As for its correctness: in all cases
 625 of the algorithm, representations of edges outside the helices is not affected by the re-writing,
 626 while edges inside the edges are accounted for by the canonical representations. ◀

627 **Proof of Proposition 5.** To start with, $G_{v \leftarrow u}$ is a minor of G , therefore $tw(G_{v \leftarrow u}) \leq tw(G)$.
 628 Then, given an optimal tree decomposition \mathcal{T} for $G_{v \leftarrow u}$, since (v, w) is an edge of this graph,
 629 there has to be a bag X containing both vertices. If $tw(G_{v \leftarrow u}) = 1$, then $X = \{v, w\}$

630 and can be split into two bags $\{v, u\}$ and $\{u, w\}$ to obtain a tree decomposition for G . If
 631 $tw(G_{v \leftarrow u}) \geq 2$, then we can simply connect a new bag $\{u, v, w\}$ and connect it to X to obtain
 632 again a valid tree decomposition for G of the same width. Therefore $tw(G) \leq tw(G_{v \leftarrow u})$ and
 633 we have the equality. \blacktriangleleft

634 **Proof of Theorem 9.** Let us distinguish two cases depending on the treewidth of G . For
 635 both of them, we consider an optimal tree decomposition \mathcal{T} of G and show how to modify it
 636 into a valid tree decomposition for the extended version of G :

- 637 ■ if $tw(G) \leq 3$ then there has to be a pair i, j ($i \leq j$) of indices $\in [1, l]$ such that $|i - j| > 1$
 638 and neither u_i, v_i or u_j, v_j are present together in one bag. Indeed, if $\forall i, j \in [1, l]$ there
 639 was such an ‘‘hop edge’’ represented, then contracting u_k, v_k together $\forall k$ would yield a
 640 clique on 5 vertices, which is forbidden if $tw(G) \leq 3$. Given such a pair i, j of indices,
 641 there has to be an edge (X, Y) of the tree decomposition that separates all occurrences
 642 of u_i, v_i from all occurrences of u_j, v_j . Let us denote $S = X \cap Y$ the separator associated
 643 to that edge. By Proposition 7, S can be assumed to be inclusion minimal, and therefore
 644 to contain exactly 2 vertices u_k and $v_{k'}$ such that $|k - k'| \leq 1$ and $i \leq k, k' \leq j$. Such
 645 a separator is depicted on Figure 3(c), as well as on Figure 7. On this latter Figure,
 646 we also depict the re-writing we perform: we introduce two new vertices x and y to
 647 the X -side of the separator, as well as intermediary bags between Y and X that will
 648 gradually transform $u_k, v_{k'}$ into x and y . To be specific, we introduce S as a bag between
 649 X and Y , and connect it to X through the series of bags $S \cup \{x\}$, $S \cup \{x, y\} \setminus \{u_k\}$,
 650 $S \cup \{x, y\} \setminus \{u_k, v_{k'}\}$ in the case (w.l.o.g) that $k \leq k'$. In addition, all occurrences of u_k in
 651 X and beyond in the subtree rooted at X and directed away from S are replaced with
 652 x and those of $v_{k'}$ with y . Since $|S| \leq tw(G)$, such a re-writing does not increase the
 653 treewidth, while representing all necessary edges for an extension of the helix by one level.
- 654 ■ if $tw(G) \geq 4$, then we consider two sub-cases depending on whether \mathcal{T} represents any
 655 ‘‘hop-edge’’ as depicted on Figure 3(d), i.e. an edge between u_k and v_l or v_k and u_l for
 656 $|k - l| > 1$. If any such edge is represented (i.e. there exists a bag containing both end-
 657 points), then by contracting the parts depicted in green on Figure 3 (d) to the extremity
 658 they contain (i.e replacing all occurrences of these vertices in the tree decomposition with
 659 their corresponding extremity), we obtain a valid tree decomposition for G_{\boxtimes} of width
 660 $\leq tw(G)$. By the inequality of Proposition 8, we get that $tw(G) = \max(4, tw(G_{\boxtimes}))$, and
 661 the extremities of the helix are a safe separator. There exists therefor an optimal tree
 662 decomposition \mathcal{T}' of G which contains S as a bag, separating the helix from the rest
 663 of the graph. By Lemma 14, replacing the sub-tree-decomposition of \mathcal{T}' corresponding
 664 to the helix with a tree decomposition for a helix longer by 1 unit does not change the
 665 width of this sub-tree-decomposition. If there is no such ‘‘hop-edge’’, then there is an
 666 edge (X, Y) in the tree decomposition that separates (u_1, v_1) from (u_l, v_l) , and to which
 667 we can apply the same re-writing as in the case of $tw(G) \leq 3$.

668 \blacktriangleleft