Accuracy of RNA Structure Prediction Depends on the Pseudoknot Grammar

Dustyn Eggers[1], Christian Höner zu Siederdissen[1,2], and Peter F. Stadler[1,3,4,5,6(✉)]

[1] Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Universität Leipzig, Härtelstrasse 16-18, 04107 Leipzig, Germany
studla@bioinf.uni-leipzig.de
[2] Theoretical Bioinformatics Lab, Bioinformatics/High-Throughput Analysis, Faculty of Mathematics and Computer Science, Friedrich Schiller University Jena, Leutragraben 1, 07743 Jena, Germany
christian.hoener.zu.siederdissen@uni-jena.de
[3] Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
[4] Institute for Theoretical Chemistry, University of Vienna, Vienna, Austria
[5] Facultad de Ciencias, Universidad Nacional de Colombia, Bogotá, Colombia
[6] Santa Fe Institute, Santa Fe, NM, USA

Abstract. Most models for pseudoknotted RNA structures can be described by multi-context free grammars (MCFGs) and thus are amenable to dynamic programming algorithms. They differ strongly in their definition of admissible structures and thus the search space over which structures are optimized. The accuracy of structure prediction can be expected to depend on choice of the MCFG: models that are too inclusive likely over-predict pseudoknots, while restrictive models by their definition already exclude more complex pseudoknotted structures. A systematic analysis of the impact of the grammar, however, is difficult since available implementations use incomparable energy parameters. We show here that Algebraic Dynamic Programming over MCFGs naturally disentangles energy models (as specified by the evaluation algebra) and the definition of search space defined by a MCFG. Preliminary computational experiments indicate that the choice of the grammar has an important impact already for short RNA sequences.

Keywords: RNA Pseudoknots · Multi-context free grammar · Algebraic dynamic programming · ADPfusion

This work was funded by the German DFG Collaborative Research Centre AquaDiva (CRC 1076 AquaDiva), the German state of Thuringia via the Thüringer Aufbaubank (2021 FGI 0009), the Carl-Zeiss-Stiftung within the program Scientific Breakthroughs in Artificial Intelligence (project "Interactive Inference"), and the German Federal Ministry of Education and Research (BMBF 031L0164C "RNAProNet").

N. M. Scherer and R. C. de Melo-Minardi (Eds.): BSB 2022, LNBI 13523, pp. 20–31, 2022.
https://doi.org/10.1007/978-3-031-21175-1_3

1 Introduction

RNA secondary structure without pseudoknots can be predicted efficiently by means of dynamic programming using a well-established standard energy model. Pseudoknots, however, play an important role in RNA function, contributing in particular to the regulation of translation and splicing, and ribosomal frameshifts [3,24]. With `pseudobase` there is a dedicated repository of biologically relevant RNA pseudoknots [26]. The RNA folding problem for general pseudoknotted structures and energy models that depend on stacked base pairs can be formally stated as follows:

General RNA Folding
Input: An ordered sequence of vertices $(x_1, \ldots, x_k)$, weights $\omega(i, j; k, l)$ that are non-positive only for "stacked edges", i.e., if $k = i+1$ and $l = j-1$, and a bound E.
Question: Is there a matching M, i.e., a set of edges such that each vertex is incident to at most one edge, with $f(M) := \sum_{\{i,j\},\{k,l\} \in M} \omega(i, j; k, l) \leq E$?

The General RNA Folding problem is known to be NP-complete if arbitrary stacking energies $\omega(\,.\,)$ can be used [1]. It remains NP complete in the unweighted case, i.e., for $\omega(i, j; i+1, j-1) = 1$ if and only if $x_i x_j$ and $x_{i+1} x_{j-1}$ are Watson-Crick base pairs [10]. Additional hardness results can be found in [20].

Several research groups proposed dynamic programming algorithms that solve the corresponding folding problem for certain restricted classes of matchings M with restrictions on the patterns of crossing edges $\{i, j\}$ and $\{k, l\}$ with $i < j < j < l$) forming the pseudoknots. These algorithms differ drastically in their definition of admissible pseudoknot types and thus in the extent of the search space, see [4,12] for an overview. The performance of the different algorithms is difficult to compare because they typically employ different parametrizations of the energy model and thus already differ in their prediction of structure without pseudoknots. It is hard to decide, therefore, whether differences in the prediction accuracy are the consequence of better energy parameters for the stems and loops of pseudoknot-free parts of the structure, or whether they have to be attributed to the pseudoknots. It has remained an open question, therefore, whether the choice of the search space has an important impact, and whether there is an optimal pseudoknot model that is sufficiently inclusive to cover the known structures but rules out structures that are impossible or unlikely to be realized at all.

In this contribution we consider re-implementations of different pseudoknot models in a common framework. This allows us, in particular, to ensure that all knot-free structures and substructures are handled identically. Furthermore, it makes it possible to assign the same energy contributions to matching types of pseudoknots. Dynamic programming (DP) algorithms are commonly defined as recursion relations that iteratively fill memo-tables. These tables are often indexed by complex structures that make the implementation of DP recursions a tedious and error prone task [6]. The theory of Algebraic Dynamic Programming

(ADP) [7] addresses this issue for a restricted class of DP algorithms for which (i) generation of the state space, (ii) scoring of states, and (iii) selection of desired solution can be separated completely. ADP is therefore the ideal framework for our endeavor, although there are attractive alternative abstract formalisms, such as "super-grammars" [18], forward-hypergraphs as an alternative description of dependencies [13] and inverse coupled rewrite systems (ICORES) [8].

2 Algebraic Dynamic Programming and `ADPfusion`

ADP utilizes a *grammar* to specify the state space and thus the structure of the recursion without any explicit reference to indices. The original setting of ADP are context-free languages, and thus productions of the form $A \rightarrow \alpha$, where A is a non-terminal and α is an arbitrary expression formed from terminals and non-terminals [7]. More recently, the formalism was extended to so-called multi-context-free grammars [16]. The main difference is that non-terminals may now be multi-dimensional, corresponding to non-overlapping sub-objects that are parsed independently. For both CFGs and MCFGs, each production determines a partition of the non-terminal on the l.h.s. Interpreting each non-terminal on the r.h.s. as a parser alleviates the need to specify indices explicitly. For instance, the simple production $S \rightarrow [\,S\,]\,S$ corresponds to the recursion relation $S_{ij} \mathrel{+}= \sum_{k=i+1}^{j} c_i S_{i+1,k-1} \bar{c}_k S_{k+1,j}$, (with the convention that the empty parse $S_{k,k-1} = 1$ serves as neutral element), where the sequence interval $[i, j]$ on which a structure "lives" is indicated by the index pair S_{ij}. The terminals c and $\bar{c}$ together signify a base pair.

In ADP, each production is interpreted by an *evaluation algebra*. Productions as grammatical objects are linked to the evaluation algebra via a common (type) signature. To understand this connection, consider the grammar $\{S \rightarrow cScS\}$, where the brackets '[' and ']' have been generalized to accommodate any particular character. The r.h.s. of the single rule of this grammar has the following "type": $c \times x \times c \times x$, while the l.h.s. holds objects that evaluate to x. The full type of the rule then is $c \times x \times c \times x \rightarrow x$. This type signature provides a constraint for both the grammar and the evaluation of parses of inputs. The terminal types c indicate that single characters are to be matched upon, while x indicates *not only* that the parse has to continue recursively *but also* that each recursive parse can immediately be replaced by a value of type x *(by means of memoization)*, for example a locally optimal score. This finally points to the structure of the evaluation algebra. An evaluation algebra is devoid of any structural notion. Instead it only contains functions that *interpret* each parse and immediately replace it by a value. In the example above, for each i, k, j, the parse $c_i S_{i+1,k-1} \bar{c}_k S_{k+1,j}$ is evaluated by a function of type, say, `Char`$\times$`Int`$\times$`Char`$\times$`Int`$\rightarrow$`Int`, which is either $S_{i+1,k-1} + S_{k+1,j} + 1$, if c_i and $\bar{c}_k$ are pairing, or $-\infty$.

The same grammar thus can be used to minimize scores, compute partition functions, density of states, or enumerate a fixed number of sub-optimal solutions by simply employing a different evaluation algebra. In addition, not only

is each algebra comparatively simple, the notion of product operations on algebras allows for easy combination of different algebras to calculate diverse and complex questions over grammars [25].

We employ a particular variant of the idea of ADP, namely `ADPfusion` [21]. This framework performs in-depth program fusion during compilation, which effectively turns a very high-level *declarative* description of a dynamic programming into tight loops that operate directly on flat memory. Authors of dynamic programs may freely mix different types of grammars, which can operate on diverse and heterogeneous index spaces [23] while still producing the desired, efficient loops that are required for dynamic programs that are asymptotically costly.

The latter property is very useful for the type of grammars we are interested in here. Multiple Context-Free Grammars (MCFGs) [19] are a particular type of weakly context-sensitive grammar that, in contrast to the general case, employ in their rewrite rules only total functions that concatenate constant strings and components of their arguments. As a consequence MCFGs admit polynomial-time parsing, i.e., the membership of word w of length n in a language generated by an MCFG G can be determined in $O(n^{c(G)})$, with a constant $c(G)$ depending only on the grammar.

Each MCFG contains rules that conform to the canonical pseudoknot-free structures – and thus substrings that are juxtaposed – and rules over substrings that contain "holes" and are interleaved with each other. The latter are represented by higher-dimensional index objects. MCFGs therefore operate on non-terminals that have an interpretation as tuples of strings over an alphabet A – rather than strings as in the case of CFGs. Due to space constraints, we cannot give a formal presentation of MCFGs here and instead refer to [16]. Instead, we use the minimal pseudoknot model of `GenussFold` [16] as a means of explaining the notations at an operational level. Consider the following productions:

$$\begin{aligned} S &\to \epsilon \mid \bullet S \mid [\, S \,]\, S \mid A_1 B_1 A_2 B_2 \\ \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} &\to \begin{pmatrix} S\,[\,A_1 \\ A_2 S\,] \end{pmatrix} \mid \begin{pmatrix} \epsilon \\ \epsilon \end{pmatrix} \qquad \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \to \begin{pmatrix} S\,[\,B_1 \\ B_2 S\,] \end{pmatrix} \mid \begin{pmatrix} \epsilon \\ \epsilon \end{pmatrix} \end{aligned} \tag{1}$$

In addition to the terminals ϵ, $\bullet$, $[\,,\,]$, which refer to the empty string, a single unpaired nucleotide and base pair, this MCFG uses three non-terminals: the one-dimensional nonterminal S represents arbitrary structures. The two-dimensional terminals $\binom{A_1}{A_2}$ and $\binom{B_1}{B_2}$ describe the two interleaved interacting parts of an H-type pseudoknot. Note that any one-dimensional index is represented by the tuple (i, j) with $i \leq j$ to fully identify a substring. A simple example of a successful parse of the string `[{]}` is given in Fig. 1.

In [16,22] we introduced a domain specific langauge (DSL) that makes it fairly convenient to write productions with 2-dimensional non-terminals. Here, we employ the same idea. First, the l.h.s. is "reformatted" such that the com-

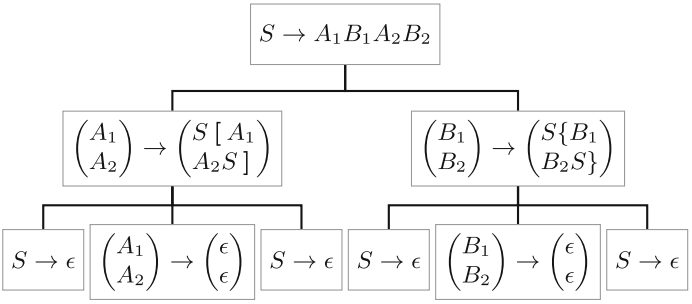

Fig. 1. Parse tree of the string `[{]}`. Compared to Eq. 1, the terminal symbols for the case $\binom{B_1}{B_2}$ have been replaced with the symbols $\{, \}$ to emphasize the two terminals forming base pairs: `[` pairs with `]`, while { pairs with }. The rule $S \to A_1 B_1 A_2 B_2$ splits the string into four (possibly empty) substrings, say, $[_{1,1}, \{_{2,2},]_{3,3}, \}_{4,4}$. The two-dimensional rule $\binom{A_1}{A_2}^{1,1}_{3,3}$ then operates on *pairs* of indices simultaneously, while the construction guarantees that only *legal* parses are derivable, i.e., the parse over $(1,1),(3,3)$ for A_1 and A_2.

ponents of the 2-dimensional non-terminal are aligned:

$$\binom{S\,[\,A_1}{A_2 S\,]} \rightsquigarrow \binom{S\ [\ A_1\ -\ -}{-\ -\ A_2\ S\ ]}$$

and then each column is transposed into a tuple to obtain a linear text

```
[S,-] [nt,-] <A,A> [-,S] [-,nt]
```

The "gap symbols" `-` are used to specify whether one-dimensional terminals and non-terminals `nt` and `S` refer to the first or second dimension. The DSL also suppresses the indices of the components of two dimensional non-terminals. One thus simply writes

```
S      ->  hpk <<< A B A B
<A,A>  ->  pka <<< [S,-] [nt,-] <A,A> [-,S] [-,nt]
<B,B>  ->  pkb <<< [S,-] [nt,-] <B,B> [-,S] [-,nt]
```

following as far as possible the notational convention of other ADP implementations [7].

Dynamic programming can be used to answer more complicated questions than the computation of maximum likelihood (or more generally score-optimal) solutions. One important class of problems concerns the relative likelihood with which a substructure occurs, weighted by its likelihood. This question, which also appears e.g. in certain algorithms for parameter fitting, requires a combination of *inside and outside* algorithms. These two algorithms describe the same search space. While the inside algorithm operates bottom-up, the corresponding outside algorithm traverses the search space in top-down order. Traditionally,

the outside algorithm is carefully constructed by hand to correctly match in all cases and generate exactly the same probabilities (or scores). It is possible to fully automate this construction [23] along with the required conversions of the each evaluation algebra. While not shown here, this automated construction is available in `ADPfusion` and thus for all grammars we consider here. This yields, for instance, algorithms to compute Boltzmann-weighted base pairing probabilities for the different classes of pseudoknotted structures.

3 Pseudoknot Grammars

The context-free grammar describing the folding algorithms for pseudoknot-free structures as implemented e.g. in the `ViennaRNA` package [9] can be written in the following form

$$\begin{aligned} S &\to \epsilon \mid \bullet S \mid BS \\ B &\to cr\bar{c} \mid crBr'\bar{c} \mid cMM'\bar{c} \\ M &\to rB \mid MB \mid M\bullet \qquad\qquad M' \to B \mid M'\bullet \end{aligned} \tag{2}$$

The non-terminals denote an arbitrary structure (S), a structure enclosed by a base pair (B), a component of a multiloop with at least one base pair inside (M), and a multiloop component whose initial base is paired (M'). The grammar conforms to the standard energy model for RNA secondary structures [27], which distinguishes hairpin-loops, interior loops (including base pairs) with a single enclosed base pair, and multiloops with two or more enclosed pairs. The terminals $\bullet$, and c, $\bar{c}$ denote an unpaired base and base pair, respectively. In addition, we write r for a region without base pairs of length at least 1 and r, r' for a pair of regions of total length at least 1. The last two lines implement the *loop decomposition*, i.e., distinguishes hairpin, interior, and multibranch loops and decomposes multiloops to support and energies that are linear in the number of unpaired bases and the number emanating stems.

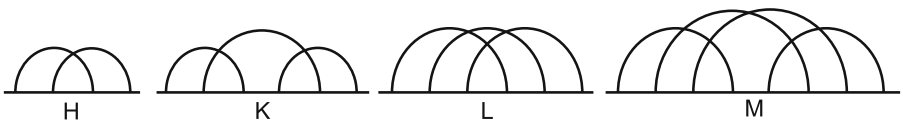

Fig. 2. The four types of pseudoknots with topological genus 1 [15] correspond to H-type pseudoknots (H), kissing hairpins (K) and two types of rare, more complex pseudoknots. The four types of pseudoknots correspond to the four alternatives in the LP^+ and RN grammars.

Many of the competing models of pseudoknots are compared in terms of their MCFG grammars and languages in [12]. Table 1 summarizes the subset considered in the contributions: The RE (Rivas & Eddy) model specifies the most

Table 1. Overview of the Pseudoknot Grammars, adapted from [4] and [12].

LP	$S \to \epsilon \mid \bullet T \mid [\,T\,]\,T \mid T A_1^{(1)} A_1^{(2)} A_2^{(1)} A_2^{(2)}$ $T \to \epsilon \mid \bullet T \mid [\,T\,]$ $\binom{A_1}{A_2} \to \binom{A_1\,[\,T}{]\,T\,A_2} \mid \binom{[\,T}{]\,T}$
LP$^+$	$S \to \epsilon \mid \bullet T \mid [\,T\,]\,T \mid T A_1^{(1)} A_1^{(2)} A_2^{(1)} A_2^{(2)} \mid T A_1^{(1)} A_1^{(2)} A_2^{(1)} A_1^{(3)} A_2^{(2)} A_2^{(3)}$ $T A_1^{(1)} A_1^{(2)} A_1^{(3)} A_2^{(1)} A_2^{(2)} A_2^{(3)} \mid T A_1^{(1)} A_1^{(2)} A_2^{(1)} A_2^{(1)} A_1^{(4)} A_2^{(2)} A_2^{(3)} A_2^{(4)}$ $T \to \epsilon \mid \bullet T \mid [\,T\,]$ $\binom{A_1}{A_2} \to \binom{A_1\,[\,T}{]\,T\,A_2} \mid \binom{[\,T}{]\,T}$
DP	$S \to \epsilon \mid \bullet S \mid [\,S\,]\,S \mid A_1^{(1)} A_1^{(2)} A_2^{(1)} A_2^{(2)}$ $\binom{A_1}{A_2} \to \binom{A_1\,[\,S}{]\,S\,A_2} \mid \binom{[\,S}{]\,S}$
RG	$S \to \epsilon \mid \bullet S \mid [\,S\,]\,S \mid A_1^{(1)} A_1^{(2)} A_2^{(1)} A_2^{(2)}$ $\binom{A_1}{A_2} \to \binom{A_1\,[\,S}{]\,A_2 S} \mid \binom{[\,S}{]\,S}$
RN	$S \to \epsilon \mid \bullet S \mid [\,S\,]\,S \mid A_1^{(1)} A_1^{(2)} A_2^{(1)} A_2^{(2)} \mid A_1^{(1)} A_1^{(2)} A_2^{(1)} A_1^{(3)} A_2^{(2)} A_2^{(3)} \mid$ $A_1^{(1)} A_1^{(2)} A_1^{(3)} A_2^{(1)} A_2^{(2)} A_2^{(3)} \mid A_1^{(1)} A_1^{(2)} A_1^{(3)} A_2^{(1)} A_1^{(4)} A_2^{(2)} A_2^{(3)} A_2^{(4)}$ $\binom{A_1}{A_2} \to \binom{A_1\,[\,S}{]\,A_2 S} \mid \binom{[\,S}{]\,S}$
AU	$S \to \epsilon \mid \bullet S \mid A_1 A_2$ $\binom{M_1}{M_2} \to \binom{M_1 K_1^{(1)}}{K_2^{(1)} K_1^{(2)} M_2 K_2^{(2)}} \mid \binom{K_1}{K_2}$ $\binom{A_1}{A_2} \to \binom{M_1}{K_1 M_2 K_2} \mid \binom{[\,S}{]\,S}$ $\binom{K_1}{K_2} \to \binom{K_1\,[\,S}{]\,S K_2} \mid \binom{[\,S}{]\,S}$

inclusive class of pseudoknots for which DP algorithms have become available so far [17]. On the other end of the spectrum, Lyngsø and Pedersen [11] considered non-recursive H-type pseudoknots. Below, we write T for the non-terminal describing pseudoknot-free structures. The LP+ model [4] includes also the four types of pseudoknots shown in Fig. 2. The model of Dirks & Pierce (DP) generalizes (LP) to include recursive H-type pseudoknots [5]. Reeder and Giegerich (RG) further restrict the appearance of unpaired bases in this setting [14]. While the two grammars look identical in Table 1, they differ in the implemention of the parsers for the terminals. This is due to a (recently remedied) limitation of the ADPfusion high-level parser that did not allow for interleaved non-terminals and terminals in the same "horizontal stack". The more efficient, original construction of RG in [14] is now possible, whereas the one in Table 1 disregards alternatives that do not fit into the RG scheme during parsing – which is semantically correct, but asymptotically suboptimal. "Simple pseudoknots" were defined by Akutsu and Uemura (AU) [1] as comprising two interleaving distinct sets of base pairs. These pairs create an interleaved stem within both groups. Base pair-

ings are organized so that the first group's right bases and the second group's left bases are arbitrarily interleaved, while the other bases are all outside the interleaved area. Categorizing secondary structures by the topological genus, Reidys et al. [15] showed that there are exactly four types of pseudoknots with genus 1, the simplest of which is the H-type pseudoknot, see Fig. 2. The genus-1 structures are referred to as (RN) below.

In order to connect the pseudoknot grammars with Turner's standard energy model [27], we interpret $[\,S\,]$ and $[\,T\,]$ as a nonterminal B in the `ViennaRNA` recursions and employ the loop-decomposition of Eq. (2). Furthermore, we use the notation $A_1^{(i)}$ and $A_2^{(i)}$ for the components of two-dimensional non-terminals that have isomorphic productions (albeit possibly with different values in the evaluations algebras). For the latter we simply dropped the superscript $^{(i)}$ in Table 1.

In line with the simplified multiloop model, we consider a single parameter, namely a *pseudoknot initialization penalty*, Ψ, which is associated with all productions that introduce a 2-dimensional non-terminal on their left side. For all helical parts within pseudoknots, the standard stacking energies are used. Unpaired positions are assigned additive contributions corresponding to the multiloop model.

4 Computational Experiments

In order to evaluate the accuracy of pseudoknot prediction we used a subset of the `RNAstrand` database [2]. Due to the computational costs of the pseudoknot algorithms, which have asymptotic running times of $O(n^6)$, we restricted ourselves to entries with at most 70 nucleotides. This leaves 131 pseudoknot-free and 63 pseudoknotted target structures.

On the pseudoknot-free subset accuracy cannot exceed the accuracy on pseudoknot-free structures.[1] Very large values of Ψ, in fact, force the predictions to be pseudoknot-free. By construction, then, there is no difference between different grammars and the `ViennaRNA`-like baseline. On this data set, we achieve a limiting F1-measure of about 0.85 for $\Psi \geq 8$ kcal/mol. We note that this value is surprisingly large in comparisons with other benchmarks of RNA folding, probably due to the short sequences.

On the subset with pseudoknots, the performance does not depend very strongly on Ψ for moderate values, it decreases, however, for large values of $\Psi > 12$ kcal/mol as sensitivity decreases. This is expected, since excessive energy penalties for pseudoknots cause them to become markedly underpredicted.

Figure 3 summarizes the results. To give a balanced picture of performance and pseudoknotted and pseudoknot-free instances despite the difference in sample sizes, we averaged the performance measure for the two samples. As expected

[1] Since we use here an energy model that is slightly simplified in the evaluation of certain loop terms compared to the full model implemented in `ViennaRNA`, occasionally we predict structures that are closer to structure model in the STRAND database and thus accuracy may also be (slightly) better than the `ViennaRNA` predictions.

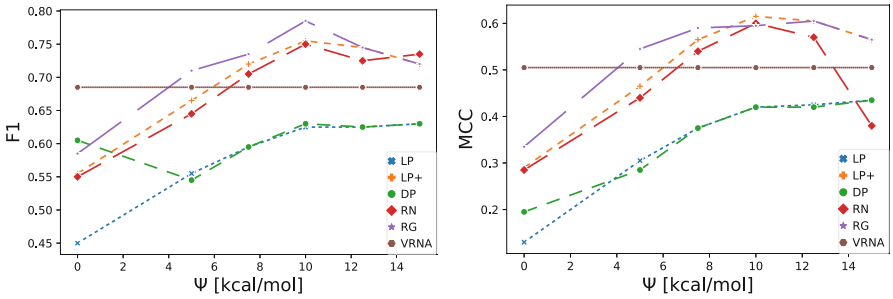

Fig. 3. Performance of five different MCFGs for pseudoknotted RNA structures in comparison with the pseudoknot-free background (VRNA). Both F1 and MCC show the best performance for $\Psi \approx 10$ kcal/mol and indicate a qualitative difference between the three grammars with recursive pseudoknots (LP+, RG, and RN) compared to DP and LP.

from analyzing the two subsets separately, we observe a performance peak for $\Psi \approx 10$ kcal/mol. Despite the short sequences in the test set we observe a superior performance of grammars that admit recursive pseudoknots.

5 Availability

This work is accompanied by git repositories. For readers who are interested in experimenting with pseudoknot grammars, we offer the "GenussFold" repository at https://github.com/choener/Prj-GenussFold. This project comes with all necessary dependencies and two options to experiment with and build pseudoknot grammars. It provides, via a `nix flake`, a complete development environment. In addition, if `nix` is not available, a more usual path via `GHC` Haskell and `cabal` is available. We refer to the readme in the project file on how to build the project. In addition, implementations for the different grammars are made available here: https://github.com/deggers/GenussFoldEnergyMin.

6 Concluding Remarks

Different grammars for the prediction of RNA structures with pseudoknots define vastly different search spaces. Variations of the grammar, therefore, include or exclude certain types of structures and thus in general will affect the predicted structures. While much effort has been expended to study and compare different implementations, no unifying framework was available in which all relevant pseudoknot model grammars are available together with a full fledged scoring system. As a consequence it has remained unclear to what extent differences in predictive power have to be attributed to the different scoring model, and to what extent the grammars themselves play an important role.

In this work we have begun a comprehensive study of the predictive power, advantages, and disadvantages due to the choice of grammar. So far, our study has been constrained to a subset of six grammars, including the pseudoknot-free RNA folding grammars from the ViennaRNA package [9]. Furthermore, we had to restrict ourselves to the set of sequences that can be folded by *all* grammars within predefined resource limits in order to accurately compare the quality of predictions. In order to minimize the influence of differences in scoring models we used here the initialization energy Ψ for a pseudoknot as the single free parameter and otherwise treated pseudoknots like multiloops. Prediction performance as a function of Ψ suggests a plausible value of about 10 kcal/mol for the optimal choice of this parameter. Interestingly, this value matches well with regression-based multiloop initialization terms, see [28] for an overview of multiloop energy models. We also observed that the two grammars LP and DP that do not admit recursive pseudoknots are outperformed by the three grammars that include recursive pseudoknots. Given the short size of the benchmarking targets this is surprising and deserves a closer examination.

This first study exposes several avenues for further exploration. When we began this study, we noted that certain production rules did not fit immediately into our framework. We chose to rewrite grammars to fit into the framework, while keeping their semantics intact. Since then, progress in ADPfusion ameliorates these shortcomings. A forthcoming more detailed study hence will encompass the full range of pseudoknot grammars. Recent improvements in the parsing and compiler fusion system further optimizes the resulting program code, enabling a systematic benchmark on significantly longer input sequences and thus more difficult instances.

Inspection of the grammars in Table 1 shows that the grammars are composed of many common rules or parts of rules. This suggests to make systematic use of another feature of ADP, namely the capability to compose grammars by additions, subtractions, and multiplications [22]. This type of construction will provide a guarantee that subsets of grammars that are supposed to be equal, will indeed generate the same structures, while at the same time reduce the complexity of the algorithms themselves. This approach will also simplify the exploration of more sophisticated energy models for pseudoknots, which in the simplest case distinguish the initialization terms for different knot types as suggested e.g. in [15].

Finally, the ability to automatically generate outside grammars opens up the possibility of calculating ensemble quantities and provides an important building block for parameter training extensions. The latter are required as grid based searches, as we performed for the pseudoknot initialization penalty, do not scale beyond two or three independent parameters.

References

1. Akutsu, T.: Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. Discr. Appl. Math. **104**, 45–62 (2000). https://doi.org/10.1016/S0166-218X(00)00186-4

2. Andronescu, M., Bereg, V., Hoos, H.H., Condon, A.: RNA STRAND: the RNA secondary structure and statistical analysis database. BMC Bioinf. **9**, 340 (2008). https://doi.org/10.1186/1471-2105-9-340
3. Brierley, I., Pennell, S., Gilbert, R.J.: Viral RNA pseudoknots: versatile motifs in gene expression and replication. Nat. Rev. Microbiol. **5**, 598–610 (2007). https://doi.org/10.1038/nrmicro1704
4. Condon, A., Davy, B., Rastegari, B., Zhao, S., Tarrant, F.: Classifying RNA pseudoknotted structures. Theor. Comp. Sci. **320**, 35–50 (2004). https://doi.org/10.1016/j.tcs.2004.03.042
5. Dirks, R.M., Pierce, N.A.: A partition function algorithm for nucleic acid secondary structure including pseudoknots. J. Comput. Chem. **24**, 1664–1677 (2003). https://doi.org/10.1002/jcc.10296
6. Giegerich, R., Meyer, C.: Algebraic dynamic programming. In: Kirchner, H., Ringeissen, C. (eds.) Algebraic Methodology And Software Technology (AMAST 2002), vol. 2422, pp. 243–257. Springer, Berlin (2002). https://doi.org/10.5555/646061.676145
7. Giegerich, R., Meyer, C., Steffen, P.: A discipline of dynamic programming over sequence data. Sci. Comput. Prog. **51**, 215–263 (2004). https://doi.org/10.1016/j.scico.2003.12.005
8. Giegerich, R., Touzet, H.: Modeling dynamic programming problems over sequences and trees with inverse coupled rewrite systems. Algorithms **7**, 62–144 (2014). https://doi.org/10.3390/a7010062
9. Lorenz, R., et al.: ViennaRNA package 2.0. Alg. Mol. Biol. **6**, 26 (2011). https://doi.org/10.1186/1748-7188-6-26
10. Lyngsø, R.B., Pedersen, C.N.: RNA pseudoknot prediction in energy-based models. J. Comp. Biol. **7**, 409–427 (2000). https://doi.org/10.1089/106652700750050862
11. Lyngsø, R.B., Pedersen, C.N.: Pseudoknots in RNA secondary structures. In: Shamir, R., Miyano, S., Sorin, I. (eds.) RECOMB 2000: Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, pp. 201–209. ACM, New York (2000). https://doi.org/10.1145/332306.332551
12. Nebel, M.E., Weinberg, F.: Algebraic and combinatorial properties of common RNA pseudoknot classes with applications. J. Comp. Biol. **19**, 1134–1150 (2012). https://doi.org/10.1089/cmb.2011.0094
13. Ponty, Y., Saule, C.: A combinatorial framework for designing (pseudoknotted) RNA algorithms. In: Przytycka, T.M., Sagot, M.-F. (eds.) WABI 2011. LNCS, vol. 6833, pp. 250–269. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23038-7_22
14. Reeder, J., Giegerich, R.: Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. BMC Bioinf. **5**, 104 (2004). https://doi.org/10.1186/1471-2105-5-104
15. Reidys, C.M., Huang, F.W.D., Andersen, J.E., Penner, R.C., Stadler, P.F., Nebel, M.E.: Topology and prediction of RNA pseudoknots. Bioinformatics **27**, 1076–1085 (2011). https://doi.org/10.1093/bioinformatics/btr090, addendum. In: Bioinformatics 28:300 (2012)
16. Riechert, M., Höner zu Siederdissen, C., Stadler, P.F. Algebraic dynamic programming for multiple context-free grammars. Theor. Comp. Sci. **639**, 91–109 (2016). https://doi.org/10.1016/j.tcs.2016.05.032
17. Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. J. Mol. Biol. **285**, 2053–2068 (1999). https://doi.org/10.1006/jmbi.1998.2436

18. Rivas, E., Lang, R., Eddy, S.R.: A range of complex probabilistic models for RNA secondary structure prediction that include the nearest neighbor model and more. RNA **18**, 193–212 (2012). https://doi.org/10.1261/rna.030049.111
19. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context free grammars. Theor. Comp. Sci. **88**, 191–229 (1991). https://doi.org/10.1016/0304-3975(91)90374-B
20. 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 2012. LNCS, vol. 7354, pp. 321–333. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31265-6_26
21. Höner zu Siederdissen, C.: Sneaking around concatMap: efficient combinators for dynamic programming. In: Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming, ICFP 2012, pp. 215–226. ACM, New York (2012). https://doi.org/10.1145/2364527.2364559
22. Höner zu Siederdissen, C., Hofacker, I.L., Stadler, P.F.: Product grammars for alignment and folding. IEEE/ACM Trans. Comp. Biol. Bioinf. **12**, 507–519 (2014). https://doi.org/10.1109/TCBB.2014.2326155
23. Höner zu Siederdissen, C., Prohaska, S.J., Stadler, P.F.: Algebraic dynamic programming over general data structures. BMC Bioinf. **16**, S2 (2015). https://doi.org/10.1186/1471-2105-16-S19-S2
24. Staple, D.W., Butcher, S.E.: Pseudoknots: RNA structures with diverse functions. PLoS Comp. Biol. **3**, e213 (2005). https://doi.org/10.1371/journal.pbio.0030213
25. Steffen, P., Giegerich, R.: Versatile and declarative dynamic programming using pair algebras. BMC Bioinf. **6**, 224 (2005). https://doi.org/10.1186/1471-2105-6-224
26. Taufer, M., et al.: PseudoBase++: an extension of PseudoBase for easy searching, formatting, and visualization of pseudoknots. Nucl. Acids Res. **37**, D127–D135 (2009). https://doi.org/10.1093/nar/gkn806
27. Turner, D.H., Mathews, D.H.: NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. Nucl. Acids Res. **38**, D280–D282 (2010). https://doi.org/10.1093/nar/gkp892
28. Ward, M., Datta, A., Wise, M., Mathews, D.H.: Advanced multi-loop algorithms for RNA secondary structure prediction reveal that the simplest model is best. Nucl. Acids Res. **45**, 8541–8550 (2017). https://doi.org/10.1093/nar/gkx512