BIOINFORMATICS ORIGINAL PAPER

Vol. 26 no. 5 2010, pages 632–639
doi:10.1093/bioinformatics/btq014

Structural bioinformatics Advance access publication January 14, 2010

Faster computation of exact RNA shape probabilities

Stefan Janssen and Robert Giegerich*

Practical Computer Science, Faculty of Technology, Bielefeld University, D-33615 Bielefeld, Germany

Associate Editor: Ivo Hofacker

ABSTRACT

Motivation: Abstract shape analysis allows efficient computation of a representative sample of low-energy foldings of an RNA molecule. More comprehensive information is obtained by computing shape probabilities, accumulating the Boltzmann probabilities of all structures within each abstract shape. Such information is superior to free energies because it is independent of sequence length and base composition. However, up to this point, computation of shape probabilities evaluates all shapes simultaneously and comes with a computation cost which is exponential in the length of the sequence.

Results: We device an approach called *RapidShapes* that computes the shapes above a specified probability threshold T by generating a list of promising shapes and constructing specialized folding programs for each shape to compute its share of Boltzmann probability. This aims at a heuristic improvement of runtime, while still computing exact probability values.

Conclusion: Evaluating this approach and several substrategies, we find that only a small proportion of shapes have to be actually computed. For an RNA sequence of length 400, this leads, depending on the threshold, to a 10–138 fold speed-up compared with the previous complete method. Thus, probabilistic shape analysis has become feasible in medium-scale applications, such as the screening of RNA transcripts in a bacterial genome.

Availability: *RapidShapes* is available via http://bibiserv.cebitec.uni-bielefeld.de/rnashapes

Contact: robert@techfak.uni-bielefeld.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on October 13, 2009; revised on December 23, 2009; accepted on January 9, 2010

1 INTRODUCTION

1.1 Motivation

1.1.1 From minimum free energy folding to abstract shape analysis Secondary structure prediction, based on thermodynamics, for RNA molecules has become an indispensable tool in RNA research, despite its well-known shortcomings (Doshi *et al*., 2004; Mathews *et al*., 1999). There are limitations in the underlying energy model (Mathews and Turner, 2006) (with respect to ion concentration, temperature and entropic effects), influences of co-transcriptional folding (Meyer and Miklós, 2004) and mechanisms such as RNA thermometers and riboswitches (Mandal and Breaker, 2004; Waldminghaus *et al*., 2009), where the prediction of an 'optimal' structure, even when correct, tells only half the story. Hence, much past and ongoing work is devoted to improving this state of affairs.

Comparative analysis of several (related) RNA sequences opens many routes for improvement (Bernhart *et al*., 2008; Havgaard *et al*., 2007; Reeder and Giegerich, 2005, only to name a few), but when studying only a single sequence, the only way forward is to give a more complete account of the folding space of the molecule. *Abstract shape analysis* is a fairly recent attempt in this direction. For shortness, here we discuss only those approaches which it is directly based upon.

The first method analyzing the complete folding space in order to assess the relevance of a secondary structure was introduced in McCaskill (1990). This approach uses the partition function to address this property. In general, the partition function provides a measure of the total number of states (structures) weighted by their individual energy at a particular temperature. For an RNA sequence s and the set $F(s)$ of all possible secondary structures for this sequence, it is defined as follows:

$$Q(s) = \sum_{x \in F(s)} e^{\frac{-E_x}{RT}} \tag{1}$$

where E_x is the energy of structure x in kcal/mol, R is the universal gas constant (0.00198717 kcal/K) and T is the temperature in Kelvin. In words, the partition function is the sum of Boltzmann weighted energies of all structures. The probability $Prob$ of a certain secondary structure $x \in F(s)$ is defined as:

$$Prob(x) = \frac{e^{\frac{-E_x}{RT}}}{Q(s)} \tag{2}$$

Intrinsic to this approach is that the probability is proportional to the (Boltzmann-weighted) energy of a structure. This approach does not provide further information on structural relevance. There is no possibility that an individual structure has a higher probability than a structure with lower free energy, and the minimal free energy (MFE) structure is always the most probable one; albeit with an individual probability that is often very close to zero. This problem has already been stated in McCaskill (1990), and the author also provides a means to alleviate it. Instead of computing the probability of a complete structure, the probabilities of atomic structural elements, i.e. base pairs, are computed by what has become known and widely used as the McCaskill algorithm.

The *RNAsubopt* program, released by Wuchty *et al*. in 1998, can give a non-heuristic enumeration of near-optimal structures. However, there is an 'embarrassingly large' (McCaskill) number of such structures in the vicinity of the energy minimum, and the problem remains how to derive significant observations from this information.

*To whom correspondence should be addressed.

In Chan *et al*. (2005) and Ding and Lawrence (2003), the authors introduced a statistical sampling algorithm that is implemented in the tool *SFOLD*. In each step of the recursive backtracing procedure, base pairs and the structural element they belong to are sampled according to their probability, obtained from the partition function. Features of the sampling procedure are that each run is likely to produce a different sample and that the same structure can be sampled multiple times, where the MFE structure has the highest probability. Still, the MFE structure is not guaranteed to be present in the sample, especially for long sequences. Sampled structures are clustered, and cluster centroids can be considered as representing the relevant structural alternatives in the molecule's folding space. The idea of producing a structure which reflects the entire distribution of structures (more properly than the MFE or maximum likelihood structure does) has been further extended by studying a variety of centroid estimators in Carvalho and Lawrence (2008), Do *et al*. (2006) and Hamada *et al*. (2009).

Abstract shape analysis (Giegerich *et al*., 2004b) implements a similar idea in a very different way. Rather than enumerating structures and clustering them thereafter, structures are classified a priori into abstract classes called shapes. Abstract shapes correspond well to human characterizations such as 'a cloverleaf structure' or 'a stem-loop with a bulge in the 5′ side'. Abstract shapes have a clean mathematical definition that allows for different levels of abstraction. In this article, we only use the most abstract level (5). Each shape describes a class of structures, and within each class, the structure of minimum free energy is defined as the *shape representative structure*, called *shrep* for short.

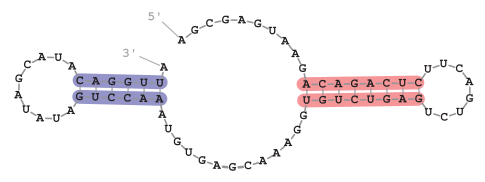
1.1.2 Computation of shape representative structures The *RNAshapes* program computes an arbitrary number of representative structures of different shapes, ranked by their energy. The top ranking shape, naturally, has the MFE structure as its represenative.

Still, knowing the energies of the shape representative structures does not tell us about the Boltzmann probability of the structures corresponding to either shape. One might naively assume that the top ranking shape also achieves the highest overall probability. This may often be true, but it does not hold in general.

Let us consider an example. We present structures in the simple dot-bracket notation made popular by the Vienna RNA package (Hofacker *et al*., 1994), where dots denote unpaired residues, and matching parentheses denote paired bases. Inspired by this notation, in abstract shapes, arrangements of complete helices are denoted by square brackets. Consider the two top ranking shapes in the following example, taken from Rfam (Griffiths-Jones *et al*., 2005):

```
>AY579432/4388-4457 from  RF00215
AGCGAGUAAGACAGACUCUUCUGCCUGAGUUCGCGGAUACAAGUGUGAAUCUAACAACGCAUACAGGUUA

Rfam family consensus
..........((((((((........))))))))...............((((((...............))))))...
```



```
Free Energy      Shape representative structure                      Shape
-14.6 .(((.((.(((((.(((..(((((........)))))....))).))..))).))..)))..........  []
-14.2 .(((((..((.((((....)))).))...)))))........(((((............)))))......  [][]
```

In this example, the second-ranking shrep of AY579432 indicates the family consensus, and hence, more likely constitutes the functional structure. However, the computed information provides no hint to this situation.

1.1.3 Computation of shape probabilities Let us now consider the shape probabilities, i.e. the accumulated Boltzmann probabilities of all structures within each shape:

```
Rank Probability Shape       Rank Probability Shape
 1)  0.7048835   [][]          6)  0.0002346   [[][]][]
 2)  0.2361259   []            7)  0.0001298   [[][][]]
 3)  0.0476083   [[][]]        8)  0.0000582   [[[][]][]]
 4)  0.0070069   [][][]        9)  0.0000028   [][][][]
 5)  0.0039475   [][[][]]     10)  0.0000025   [[][[][]]]
```

Such a calculation shows that shape `[][]` is more populated than shape `[]` and dominates the Boltzmann ensemble. Furthermore, this analysis shows that structures of all other shapes ranked 3 or above are unlikely to play a functional role.

Computation of shape probabilities is implemented in the second release of the *RNAshapes* program (Steffen *et al*., 2006; Voß *et al*., 2006).

1.1.4 Properties and uses of shape probabilities Compared with plain folding energies, shape information has several advantages: shape probabilities are independent of sequence length and base composition, and hence are meaningful across related sequences from different organisms. For example, shape probabilities have been used to assign significance levels to predicted miRNA precursors in large-scale studies (Berezikov *et al*., 2006; Lu *et al*., 2008). Consensus shapes for a set of sequences can be determined quickly and have been used as the basis of consensus structure prediction (Reeder and Giegerich, 2005). Finally, it has been determined that Rfam families can be indexed by their shape spectra, which leads to a significant speed-up of Rfam searches (Janssen *et al*., 2008).

1.1.5 Computational cost of probabilistic shape analysis While standard MFE folding algorithms have the asymptotic runtime of $\mathcal{O}(n^3)$, where n is the sequence length, probabilistic shape analysis has a runtime of $\mathcal{O}(rn^3)$. The value r is the (expected) number of *all* shapes encountered in the folding space of the analyzed RNA sequence. The aymptotic number of *all* shapes of *all* sequences of length n has recently been determined to be $1.20^n \cdot 5.13 \cdot n^{\frac{-3}{2}}$ (Lorenz *et al*., 2008; Nebel and Scheid, 2009), but the expected number of shapes encountered for an individual sequence is not known. Rudimentary measurements in (Voß *et al*., 2006) indicate a value of $r \approx 1.1^n$. As a consequence of this exponential factor, probabilistic shape analysis for a sequence of length 400 requires ∼11.5 h and 2.7 GB memory on present day hardware.

It is clear that when computing the probabilities of *all* shapes, a factor related to the number of shapes cannot be avoided. But in practice, we are interested only in a handful of top-ranking shapes. We can compute the top k shapes ranked by shrep energy in $\mathcal{O}(kn^3)$ time—so it may seem surprising that this should not be possible for shapes ranked by probability. The explanation is that shape probabilities add up from many small contributions, and there is no way to determine early which subshapes may later contribute to the top-ranking ones. In other words, Bellman's Principle of Optimality (Giegerich *et al*., 2004a), the prerequisite of dynamic programming algorithms, does not apply for shape probability computation.

There may be a different approach to compute shape probabilities in polynomial time, but the chances do not look good: problems of this type are closely related to the path labeling problem for hidden Markov models, shown to be NP-hard in Brejová *et al.* (2007).

For this reason of computational expense, the *RNAshapes* program also provides two heuristics: one is a low probability filter that excludes subshapes of very low probability, e.g. $<10^{-6}$. This implies that the overall partition function appears smaller than it is. The other heuristic is a sampling mode, akin to *SFOLD*, but defining clusters a priori as shapes. Still, sampling gives up on computing exact probabilities, and also becomes expensive for longer sequences when a large number of samples must be drawn. Therefore, we set out here to provide a runtime heustistic to compute the shapes of highest probability. A *runtime* heuristic means that we still compute the exact probabilities, efficiently in many cases, but with no guarantee for polynomial runtime in general.

1.2 Outline of ideas

Let s be the sequence under consideration, and length$(s)=n$. The partition function $Q(s)$ of the complete folding space $F(s)$ can be computed efficiently in $\mathcal{O}(n^3)$ time. Now consider a specific shape p, such as `[][]`. All the structures of shape p constitute a subspace $F_p(s)\subset F(s)$. The probability of p is

$$Prob(p,s)=Q_p(s)/Q(s), \text{ where } Q_p(s)=\sum_{x\in F_p(s)} e^{\frac{-E_x}{RT}} \qquad (3)$$

This means that $Q_p(s)$ is itself the partition function of $F_p(s)$, and with a special program that folds s exactly and only into the structures of $F_p(s)$, we can compute $Q_p(s)$ efficiently in $\mathcal{O}(n^3)$ time.

A program folding an RNA sequence into a restricted set of structures, using the standard energy model, is called a thermodynamic matcher (TDM) (Reeder and Giegerich, 2005). Such TDMs are produced, for example, via the tool *Locomotif*, where a user composes pictures of annotated RNA structures, which are then compiled into programs for RNA motif search (Reeder *et al.*, 2007). We use a similar TDM generator which, given an abstract shape p, generates the TDM for p, which computes the partition function.

The overall idea is as follows:

(1) We compute $Q(s)$ in $\mathcal{O}(n^3)$.

(2) We enumerate (heuristically) a series of promising shapes $p_1, p_2, \dots$.

(3) For each p_i, we generate TDM$_i$ as a program coded in algebraic dynamic programming (ADP) style (Giegerich *et al.*, 2004a).

(4) We compile TDM$_i$ and execute it to compute $Q_{p_i}(s)$ in $\mathcal{O}(n^3)$, and obtain $Prob(p_i,s)$ according to Equation (3).

(5) We continue until a specified portion of the shape probabilities is exhausted.

Since the time for TDM generation and compilation is small compared with their execution time, overall runtime of this heuristics is $\mathcal{O}(tn^3)$, where t is the number of TDMs (shapes) used. We call this approach *RapidShapes*.

2 A METHOD FOR FASTER SHAPE PROBABILITY COMPUTATION

2.1 Basic problem: shapes with a least $T\%$ probability

Here, we define the standard problem we want to solve efficiently; later we shall discuss variations of it. Let us set up a probability threshold T for the shapes of interest, say 0.9 or 0.6 as used in Berezikov *et al.* (2006), or maybe as low as 0.1. Given T, only a constant number of shapes (1 with the first two settings, 9 for the third or $\lceil(1/T)-1\rceil$ in general) can meet the threshold—independent of the sequence length n. Note that there may be no shapes at all meeting the threshold. On the other hand, there is a large population of shapes living in sub-thresholdia, their number growing exponentially with n.

2.1.1 Problem definition Given an RNA sequence s of length n and a threshold $0<T\leq 1$, compute all shapes p of s with $Prob(p)\geq T$, together with their shape representative structures and free energy.

This definition permits that some shapes with subthreshold probability will also be computed, but the goal is, of course, to minimize our efforts spent on those. We have to solve two subproblems, namely the analysis of subspaces $F_p(s)$ and the generation of a good list of 'promising' shapes.

2.2 Analysis of the folding space partitioned by shape

A TDM folds a sequence only to a restricted set of structures. For *RapidShapes*, such a restricted set of structures comprises all structures of a particular shape (and no other). A TDM can compute the partition function value $Q_p(s)$ of its restricted folding space $F_p(s)$ in $\mathcal{O}(n^3)$ time, just as the unrestricted RNA folding program does for the complete folding space $F(s)$. Since $Q_p(s)$ is the sum of *all* structures in $F_p(s)$, and $F_p(s)$ is a precise subset of $F(s)$, we have to ensure that a TDM folds exactly those structures constituting the shape class p. Our strategy is to generate such programs on demand.

2.2.1 Representing structures and shapes as grammars RNA structures are conveniently described by context-free grammars (Durbin *et al.*, 1999), where the parses of an RNA sequence s indicate all its possible foldings $F(s)$. Parses implicitly assign a score with each structure, be it probabilities from a stochastic model, base pair counts or thermodynamic energies. Here, we use tree grammars to describe structures. Tree grammars make explicit the semantics of each grammar rule, and can be compiled directly into executable code using the ADP technology (Giegerich *et al.*, 2004a).

An expository simplification of the tree grammar that we use to compute $F(s)$ is given in Figure 1. The left part of Figure 2 shows one of the candidate structures, tree t_1, which this grammar assigns to the input sequence `ACCAAAGG`. The operators `root`, `last`, `stack` and `hairpin` will be used to compute all relevant properties of this structure candidate. The actual grammar used in *RNAshapes* as well as in *RapidShapes* uses 26 rules and 35 operators to accomodate the thermodynamic energy model, including dangling bases.

We can imagine the progress of an RNA folding program based on tree grammars in two phases. Phase one is the generation of all candidate structures (trees), while in the second phase a score is assigned to each candidate. The score of a candidate depends on the scheme that describes how the tree operators must be evaluated in phase two. For example, if we want to assign a Vienna Dot-Bracket representation as a *score* to t_1, we can apply $\mathcal{A}_{\text{dotBracket}}$

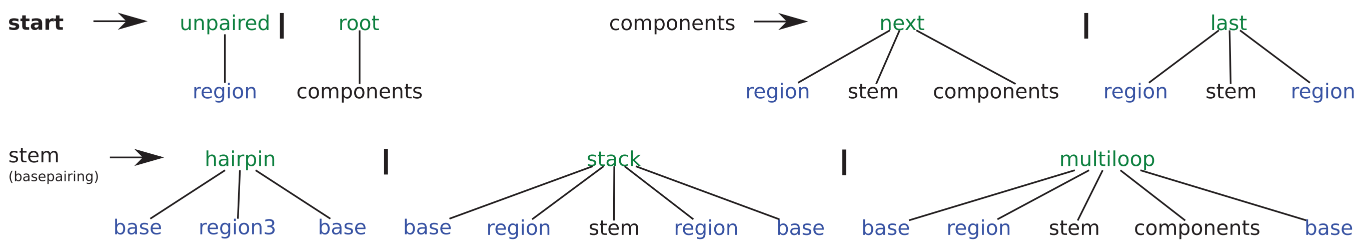


Fig. 1. Tree grammar for folding all RNA structures G_{all}: terminals are colored in blue, where `base` is a single unpaired base, `region` is a possibly empty stretch of unpaired bases and `region3` has at least three unpaired bases. Non-terminals are black. The axiom **`start`** is written in boldface. The operators are given in green. The subscript `basepairing` indicates that the outmost bases have to form a base pair. All structures are either `unpaired` or consists of one (`last`) or many (`next`) consecutive `stems`. A stem can be a `hairpin` or a `multiloop`, which both can be extended by a `stack`. `stack` includes internal loops, bulges and stacked base pairs, depending on whether two, one or none of both `regions` are empty.

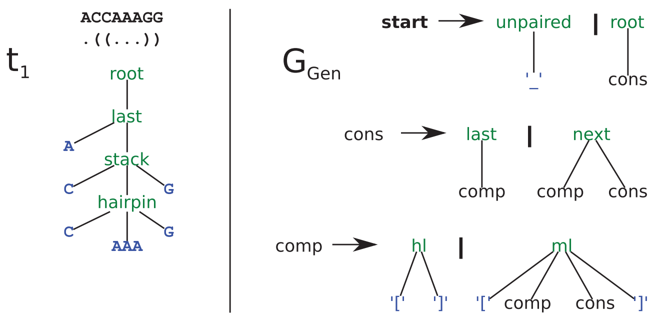


Fig. 2. Left: parse t_1 as one of many possible parses for `ACCAAAGG` with G_{all}. The Vienna Dot-Bracket representation is `.((...))` and the shape string of level 5 is `[]`. Right: grammar G_{Gen} to generate specialized tree grammars: operators, non-terminals and terminals are coloured as in Figure 1. Instead of an RNA sequence, the input for this grammar is a shape string.

Table 1. Equations for the two evaluation schemes $\mathcal{A}_{\text{dotBracket}}$ in the second column and $\mathcal{A}_{\text{shape5}}$ in the last column

Operator	$\mathcal{A}_{dotBracket}$	$\mathcal{A}_{shape5}$
root$(x)=$	x	x
last$(l,x,r)=$	$\ldots_{\vert l\vert}+x+\ldots_{\vert r\vert}$	x
is$(x)=$	x	x
sr$(a,l,x,r,a')=$	$(+\ldots_{\vert l\vert}+x+\ldots_{\vert r\vert}+)$	x
hl$(a,m,a')=$	$(+\ldots_{\vert m\vert}+)$	`[]`
unpaired$(u)=$	$\ldots_{\vert u\vert}$	_
next$(u,x,y)=$	$\ldots_{\vert u\vert}+x+y$	$x+y$
ml$(a,u,x,y,a')=$	$(+\ldots_{\vert u\vert}+x+y+)$	`[`$+x+y+$`]`

$+$ is string concatenation and $\ldots_{|m|}$ means one '.' for each base in m. a and a' are the two partners of a base pair. The resulting Vienna Dot-Bracket string for t_1 would be `.((...))`. Since G_{all} is semantically unambiguous with respect to $\mathcal{A}_{\text{dotBracket}}$, there is only one structure that can form this specific Vienna-Dot-Bracket representation. The shape representation for t_1 is '`[]`'. Since $\mathcal{A}_{\text{shape5}}$ does not account for unpaired regions or the number of base pairs in a stack, many structures, generated by G_{all}, will have the same shape representation.

to get the string `.((...))` (Table 1). By exchanging $\mathcal{A}_{\text{dotBracket}}$ with $\mathcal{A}_{\text{shape5}}$, we get `[]` for t_1. Other schemes are used to calculate thermodynamic energies, probabilities or base pair counts.

2.2.2 Generating grammars from shapes Given shape p, we generate the specialized grammar G_p. We do so using another tree grammar G_{Gen}, see right part of Figure 2, which parses a shape representation as its input. The rules of G_{Gen} follow the intention of the shape abstraction: the allowed submotifs (`comp`) are either adjacent helices (`cons`), or embedded helices within multiloops. An exception is the completely unpaired structure (`unpaired`).

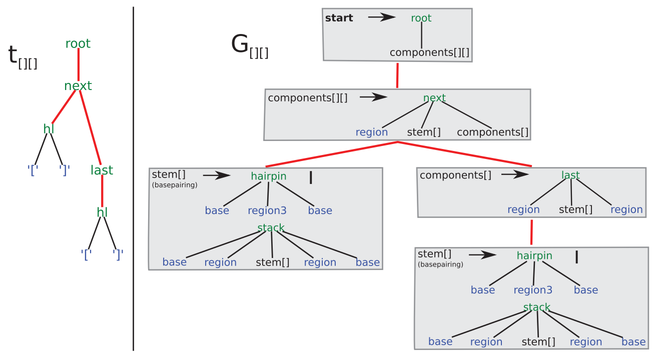


Fig. 3. TDM for the shape `[][]`: the left side shows $t_{[][]}$, the only parse of '`[][]`' with G_{Gen}. On the right side is the generated grammar $G_{[][]}$ with five rules depicted that is constructued by applying the evaluation scheme $\mathcal{A}_{\text{Gen}}$ to the operators of $t_{[][]}$. To see the similarities the backbone of both trees are colored in red.

For our example $p=$'`[][]`' (see left part of Fig. 3), $t_{[][]}$ is the unique parse of `[][]` with G_{Gen}. The restricted tree grammar $G_{[][]}$ for folding structures of shape class `[][]` is constructed by applying a grammar-generating evaluation scheme $\mathcal{A}_{\text{Gen}}$ to $t_{[][]}$. The details of this generation exceed the scope of the present article. Equations for $\mathcal{A}_{\text{Gen}}$ are available in the Supplementary Material. The result is shown in the right part of Figure 3. The important difference to G_{all} is that the non-terminals are now position specific—or better helix specific—by the addition of suffices to their names. Furthermore, the alternatives are reduced for some positions. For example, the rule `components[][]` lacks the `last` alternative, while `components[]` lacks `next`.

Since a grammar is a *set* of rules, the identical rules in Figure 3, namely `stem[]`, can be handled by the same matrices in the dynamic programming code, which saves some memory space as well as runtime.

2.2.3 Using the standard energy model Although $Q(s)$ and $Q_p(s)$ are computed by two independent programs, they have to fold the same structures *and* evaluate them to the same energy values, i.e.

Boltzmann weighted energies. We can assure this by using the identical evaluation scheme for both programs. This comes for free, because the operators in G_{all} and G_p are always the same.

THEOREM 1. $G_{\text{Gen}}(\mathcal{A}_{\text{Gen}},p)$ generates a TDM_p which correctly computes $Q_p(s)$.

PROOF. By construction, TDM_p recognizes unambiguously all structures of shape p. When applied to s, it exactly constructs $F_p(s)$ and hence computes $Q_p(s)$. That our implementation actually satisfies this mathematical property, can be systematically tested by checking $\sum_p Q_p(s) = Q(s)$.

2.3 Heuristic shape selection

To run *RapidShapes* for a given sequence s, we construct a list $L(s)$ of 'promising' shapes. For each shape $p \in L(s)$, we construct TDMs and compute $Prob(p,s)$ in $\mathcal{O}(n^3)$ time. Ideally, $L(s)$ would only contain the shapes above the threshold, but this is exactly the problem to be solved.

2.3.1 Selection by shrep energies The simple shape analysis for a given sequence s computes the top k shapes ranked by the energy of their shreps (cf. Section 1). Although shape ranks by shrep energy and shape ranks by probability do not agree, there is a positive correlation between shrep energies and shape probabilities. We start with a small k to compute $L(s) = [p_1,\dots,p_k]$ by simple shape analysis. If $\sum_{i=1}^{k} Prob(p_i,s) < 1-T$, we repeat the simple shape analysis with larger k to extend $L(s)$.

2.3.2 Selection by sampled frequencies Sampling of structures to estimate shape probabilities can be done very fast, but the results are not exact. However, the reported shapes may be the most likely ones, in spite of their probabilities being incorrect. To combine both advantages, we use the quickly calculated shapes from the sampling as members for $L(s)$ and precisely determine their probabilities via TDMs. Sampling requires a bound on the number of samples drawn, which was set to 1000 for this study. This might overlook shapes with probability $\geq T$, but this chance can be decreased by drawing more structures.

2.4 Asymptotics

To satisfy the problem definition, given in Section 2.1, *RapidShapes* must calculate the probability, the shrep structure and its free energy for all k shapes with $Prob(p) \geq T$. All three values can be computed for one shape by using specialized evaluation schemes with the TDM, where $Prob(p)$ is $Q_p(s)$ divided by $Q(s)$, which must be calculated just once. For the sake of speed, we seperate the computation of shape probabilities for all shapes in $L(s)$ from the computation of shrep structures and free energy values for all k shapes, with $Prob(p) \geq T$, because usually $|L(s)| >> k$. This leads to an asymptotic runtime of $\mathcal{O}(n^3 + |L(s)| \cdot n^3 + k \cdot n^3 + l)$.

3 EVALUATION

3.1 Evaluation setup

Our evaluation uses a random and a 'real' test set. The random testset is constructed to uniformly cover a sequence length of 5–1000 nt in steps of 5 nt. We use two sequences for each length, so the test set contains 400 sequences. The realistic test set contains 1092 candidates from a recent experimental screen for bacterial non-coding RNAs (Schlüter,J.-P. *et al.*, manuscript in preperation).

3.1.1 Oracle To mark the theoretical maximum speed up for *RapidShapes*, we assume an *oracle* that a priori denominates the shapes for $L(s)$, ordered by their shape probabilities. This would allow to use the minimum number of TDMs for any choice of T. Of course, such an oracle does not exist, but we can determine what it would have returned by dropping subthreshold shapes from $L(s)$ after their evaluation.

3.2 Results on random data

The performance evaluation of *RapidShapes* has two aspects: the effective number of shapes that have to be evaluated, and the absolute gain in runtime.

3.2.1 Required number of TDMs Figure 4A is a comparison of the growth of $F(s)$ and $L(s)$ for different methods to fill this list with promising shapes.

As expected, just a very small number of all existing shape classes in $F(s)$ (cyan colored curve) seems to account for a major part of all structure probabilities. The number of TDMs, which must be generated, compiled and executed for *RapidShapes*, is strikingly smaller than $|F(s)|$, regardless of the method for filling $L(s)$.

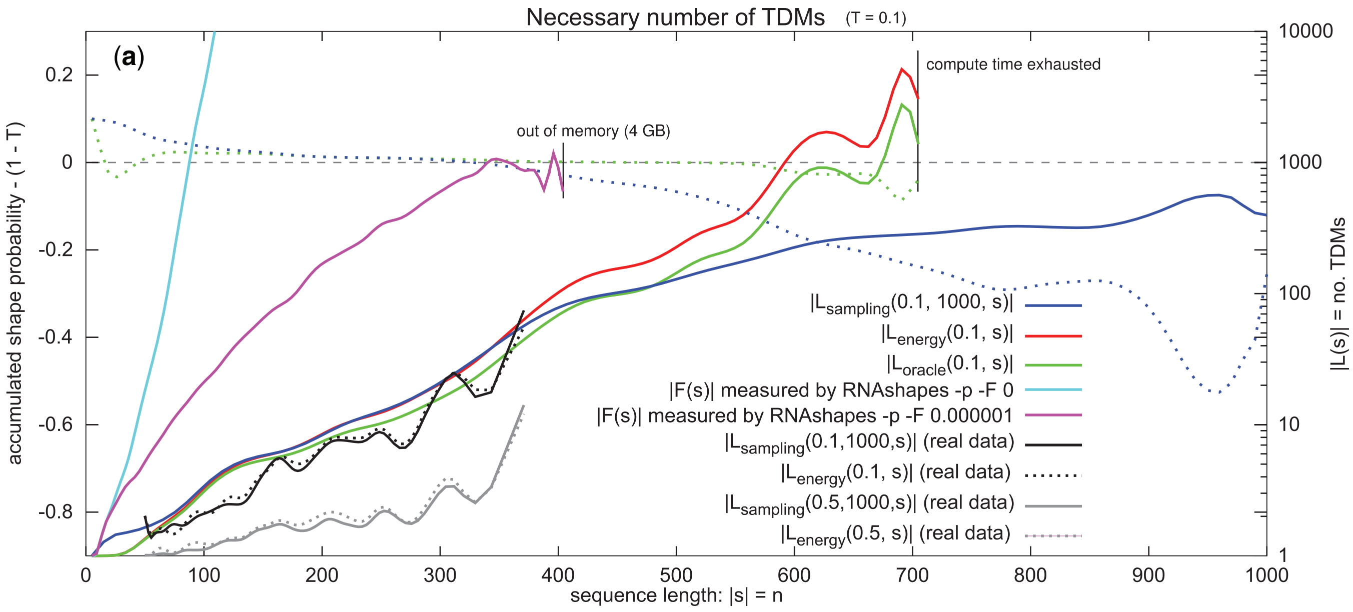
The *Selection by shrep energies* (red curve) is not perfect, because the green curve of the omniscient oracle is somehow below, but the distance is not too wide and it follows the trend of the oracle.

Selection by sampled frequencies (blue curve) runs the risk of not creating enough shapes, due to the limited sample size. Sampling 1000 structures can result in at most 1000 different shapes, and normally much less. For larger sequences the accumulated probability of these shapes may not be sufficient to cover $1-T$. Where this strategy needs even less TDMs than the oracle, it comes with an increasing proportion of unexplored parts of $F(s)$. The amount of unaccounted folding space for the *Selection by sampled frequencies* method is indicated by the blue dotted curve in Figure 4A.

It is interesting that the sampling strategy becomes faster than the energy-based strategy exactly where it starts missing shapes above the threshold. We conclude that there lies no real advantage in the sampling strategy when computing *all* exact shape probalities above the threshold is required.

3.2.2 Runtime speed-up Different shape strings result in different TDMs, i.e. different grammar sizes. The larger the grammar, the higher is the runtime for generation, compilation and execution, independent of the input sequence. To include these effects into the evaluation results, Figure 4B shows our analysis of empirically measured runtimes.

While computing a probabilistic shape analysis, the program *RNAshapes* uses a filter to exclude all subshapes with a very low probability ($\leq$0.000001 by default), although they might contribute to highly probable shapes. This means the results are no longer exact, but this threshold is wisely chosen. The error for the probability values is very small and the speed-up is significant. Without the filter, *RNAshapes* exceeds memory for sequences as small as 130 nt (cyan colored curve in Fig. 4B). With activated filtering, valid input sequences can have up to 400 nt (magenta curve). This comes with a



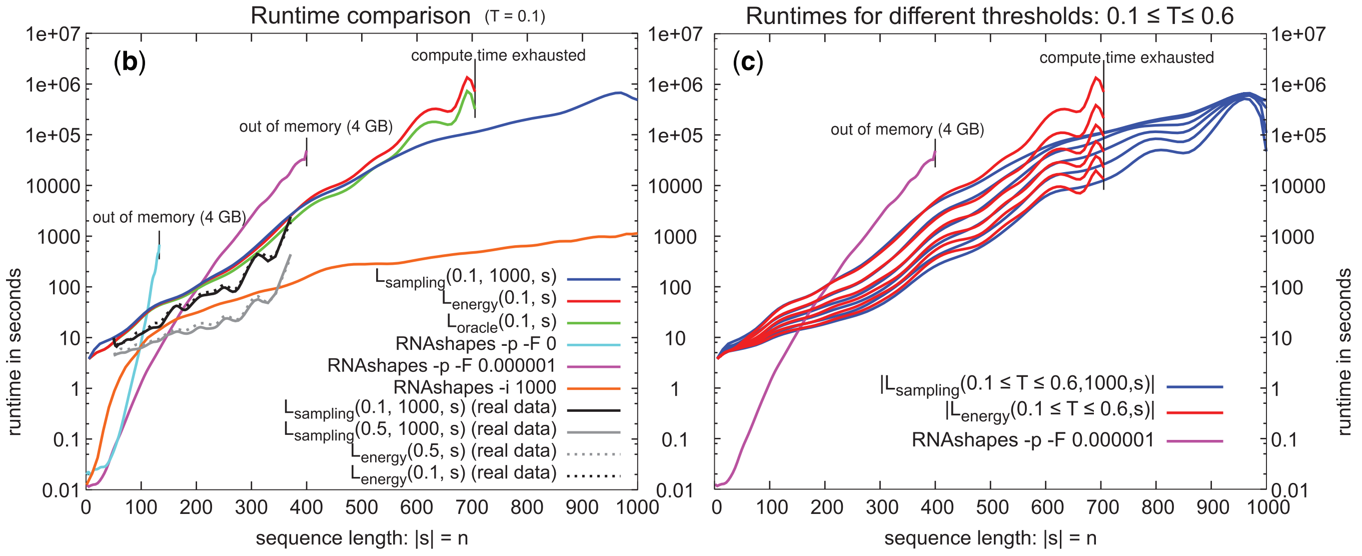


Fig. 4. Evaluation results: (**A**) necessary number of TDMs compares the growing numbers of shapes in $F(s)$ and $|L(s)|$, which is the number of shapes used in *RapidShapes*. The x-axis is sequence length, the logarithmic scale right y-axis is $|L(s)|$. $F(s)$ is measured by *RNAshapes* with low-probability filter set to 10^{-6}, colored in cyan, and without filtering, colored in magenta. We tested three different methods to fill $L(s)$ for $T=0.1$. The green colored $L_{\text{oracle}}(0.1,s)$ is the optimal choice of shapes for *RapidShapes*. 'Selection by shrep energies' $L_{\text{energy}}(0.1,s)$ is colored in red and 'Selection by sampled frequencies' $L_{\text{sampling}}(0.1,1000,s)$ for 1000 samples per sequence is colored in blue. The previous curves result from the random data, while the black ($T=0.1$) and gray ($T=0.5$) curves arise from the real dataset, when applying the 'Selection by sampled frequencies' method. 'Selection by shrep energies' for the real dataset is shown by the dashed black and gray curves. The left y-axis depicts the amount of unaccounted folding space, i.e. the accumulated probability of all evaluated shapes minus $(1-T)$. The dotted blue curve corresponds to $L_{\text{sampling}}(0.1,1000,s)$, while the dotted green curve result from the most probable 1000 shapes, which are too few to cover $1-T$ for sequences longer than ~550 bases. (**B**) Runtime comparison illustrates actual runtimes of the various methods instead of counting shapes. The logarithmic scale y-axis is here runtime in seconds. The newly introduced orange curve depicts the runtime for the pure sampling process via *RNAshapes* of 1000 structures. (**C**) Runtimes for different thresholds show the influence of selecting different values for the threshold T. All curves are smoothed via the gnuplot option 'smooth bezier'. For more details on the curves, consider the online-only Supplementary Material. We stopped our evaluation at sequence length 700 after consuming 574 CPU days.

mean squared error between true and calculated shape probabilities of 4.62×10^{-11} (tested with 750 sequences of both testsets < 130 bases).

For short sequences, the overhead of constructing TDMs prevents *RapidShapes* from being useful, but with input sequences > 208 nt there is a growing speed-up, compared with *RNAshapes*. Furthermore, the heuristic makes it possible to compute shape probabilities for sequences > 400 nt, where *RNAshapes* runs out of memory.

The achieved speed-up depends on the choice of threshold T. Our previous measurements were made for a rather low threshold of $T = 0.1$. In practice, a threshold of 0.6 or even 0.9 makes sense when checking for the existence of a dominant shape. The larger the T, the faster is *RapidShapes*—this is demomstrated by the different red and blue curves in Figure 4C for 'selection by shrep energies' and 'selection by sampled frequencies', respectively.

3.3 Results on real data

Natural RNAs, whether functional or not, are not random sequences. Functional non-coding RNAs are known to be optimized for good folding energy, although this signal is not strong enough to discern functional from non-functional RNA. All natural RNA has a bias toward lower folding energy than random sequences. (Cf. Clote *et al.*, 2005, and the long debate summarized therein). We can expect this bias to favor a small number of shapes with a high probability over a more even distribution in real RNAs. This fact must lead to *RapidShapes* requiring a smaller number of TDMs and hence becoming faster. This result is confirmed on our real data testset. It consists of 1092 candidates from a bacterial screen for non-coding RNAs, identified via deep sequencing and microarray expression analysis. Returning to Figure 4A, consider the black and the gray line, computed with $T = 0.5$ and $T = 0.1$, respectively. The smaller number of required TDMs also results in a moderate speed-up. Out of these 1092 sequences, are 98 >208 nt, the break-even point (for $T = 0.1$) where *RapidShapes* starts to become faster than *RNAshapes*. While *RNAshapes* with activated filtering consumes 67.5 h to calculate shape probabilities, *RapidShapes* takes only 7.3 h of runtime.

4 DISCUSSION

4.1 Speed-ups and brake-even points achieved

Abstract shape probabilities, as computed by *RNAshapes*, provide useful information beyond MFE folding. Due to the large computational cost, previously, a sampling heuristic had to be used for sequences longer than about 200 bases. This heuristic, however, has the disadvantages that (i) it does not return exact probabilities, (ii) does not account for the part of the folding space not covered by the sampling and (iii) does not return shape representative structures. The mean squared error between sampled and exact shape probabilities for all test sequences is 1.64×10^{-4}. In a case where a particular shape clearly dominates all others, these disadvantages do not really matter, as this shape will be sampled many times (dwarfing the probabilities of other shapes), and the shape representative structure has a good chance to be in among the sample. However, one cannot know beforehand whether this situation applies.

The approach *RapidShapes* presented here overcomes these limitations and enables the computation of shape probabilities a much wider sequence range. It is a *runtime* heuristic—i.e. the computed probabilities are exact, we obtain the shape representative structures, and we know about the uncovered amount of probability in the folding space. What cannot be guaranteed is polynomial runtime in a strict asymptotic sense, but the evaluation shows that *RapidShapes* performs well in practice.

.5The speed-up achieved by *RapidShapes* depends on the threshold and on the sequence length. Using a very relaxed threshold $T = 0.1$, *RapidShapes* becomes faster than the traditional method (using its low-probability filter) at sequence length 208 nt. At sequence length 400 nt, *RapidShapes* is faster by a factor of 10. Taking a more stringent threshold at $T = 0.6$, *RapidShapes* becomes faster at sequence length 159 nt, and at sequence length 400 nt, the speed-up factor is about 138. Independent of threshold, *RapidShapes* is the only practical method to compute exact shape probabilities for sequences > 400 nt.

4.1.1 Expected numbers of shapes above a probability threshold The combinatorics of shapes has found considerable interest recently, but the expected number of shapes of a sequence of length n is still unknown. Our large-scale evaluation has produced some empirical data in this respect. Assuming a simple exponential growth pattern of $\mathcal{O}(\alpha^n)$, we can estimate α. Our data (Fig. 4A) suggest that $\alpha = 1.096439^n$ for the number of all shapes of a sequence of length n (cyan curve), $\alpha = 1.020742$ for all shapes with a probability $> 10^{-6}$ (magenta curve), and $\alpha = 1.011290^n$ for all shapes with probability larger than $T = 0.1$ (green oracle). *RapidShapes* computes $\alpha = 1.011341^n$ shapes (red curve) and hence is quite close to optimal. For $T = 0.5$, the oracle value is $\alpha = 1.005604$ (data not shown).

4.2 Problem variants

4.2.1 ***k** best shapes* As explained initially, we can efficiently compute the k best shapes of sequence s ranked by shrep energy, but not ranked by probability. In order to compute the k best shapes ranked by probability, we compute shape probabilities according to the L_{energy} strategy. Assuming $K \geq k$ shapes have been computed, and $p_1,\dots,p_k$ are the best k shapes seen so far, we can stop computation as soon as $1 - \sum_{i=1}^{K} Prob(p_i) \leq Prob(p_k)$.

4.2.2 Best shape only If we ask for the best shape no matter how small its probability is, we can do no better than to apply the above strategy for $k = 1$. However, if we are interested in the existence of a *dominant shape*, loosely defined here as one which is more likely than all the rest together, we just solve the standard problem with threshold $T = 0.5$.

4.3 Implementation alternatives

4.3.1 Algorithm parameterization versus generation From an algorithmic point of view, or method of generating, compiling and running algorithms for subproblems on the fly appears somewhat unusual. As an alternative, one could think of modifying the code of the traditional method to accomodate a target shape p as an extra parameter, and restrict the folding to structures which match p running through list L_{energy}. This would turn the general method into the equivalent of a TDM for p. However, this would slow down the inner loop of an $\mathcal{O}(n^3)$ algorithm, whereas generating and compiling takes $\mathcal{O}(n)$ time (empirically: less than 7.76% of the overall process) to yield an $\mathcal{O}(n^3)$ TDM algorithm without such a slowdown. And besides, generating TDMs from shapes has other applications, e.g. in RNA motif search.

4.3.2 Other shape enumeration heuristics We have experimented with other ideas of enumerating promising shapes, such as (i) using a precompiled library of frequently encountered shapes, or (ii) always computing low complexity shapes (such as `[]`) first. Neither of these ideas has provided an improvement. When computing shape probabilities for abstraction levels $i<5$, a good strategy may be to first compute best shapes of level $i+1$ and then computing their subshapes at level i. This is possible since shape abstraction levels form a perfect hierarchy. However, this idea has not been further explored yet.

4.4 Open problems

Given a particular TDM, it is easy to generate a scanning version to find high probability instances of its shape in a longer sequence. An adaptive window size, subject to a reasonable upper bound, also seems feasible. However, thinking of for RNA gene prediction based on dominant shapes, we would need a scanning version that dynamically changes the shape as it moves along the sequence. This presents a challenge for future research.

Our technique does not depend on the concrete information that is accumulated for each shape. Recent approaches such as *CONTRAfold* (Do *et al*., 2006) and *CG* (Andronescu *et al*., 2007) replace the classical thermodynamic model by stochastic models, trained from structural data via machine learning techniques. To benefit from our aproach, these methods need to be augmented to support shape abstraction. Technically, they are based on different grammars than *RNAshapes*. For these grammars, shape abstraction functions need to be defined and implemented. Then, our TDM generator and the strategies described here should carry over without change.

ACKNOWLEDGEMENTS

The authors thank Jens Reeder for the TDM generator from *Locomotif*, Georg Sauthoff for the ADP to C code generator (GAPc), Robert Homann for his program *randomseq* and Stefanie Schirmer for careful proofreading and discussions.

Conflict of Interest: none declared.

REFERENCES

Andronescu,M. *et al*. (2007) Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, **23**, i19–i28.

Berezikov,E. *et al*. (2006) Many novel mammalian microRNA candidates identified by extensive cloning and RAKE analysis. *Genome Res*., **16**, 1289–1298.

Bernhart,S.H. *et al*. (2008) RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinformatics*, **9**, 474.

Brejová,B. *et al*. (2007) The most probable annotation problem in HMMs and its application to bioinformatics. *J. Comput. Syst. Sci*., **73**, 1060–1077.

Carvalho,L.E. and Lawrence,C.E. (2008) Centroid estimation in discrete high-dimensional spaces with applications in biology. *Proc. Natl Acad. Sci. USA*, **105**, 3209–3214.

Chan,C.Y. *et al*. (2005) Structure clustering features on the Sfold Web server. *Bioinformatics*, **21**, 3926–3928.

Clote,P. *et al*. (2005) Structural RNA has lower folding energy than random RNA of the same dinucleotide frequency. *RNA*, **11**, 578–591.

Ding,Y. and Lawrence,C.E. (2003) A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res*., **31**, 7280–7301.

Do,C.B. *et al*. (2006) CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, **22**, e90–e98.

Doshi,K.J. *et al*. (2004) Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for RNA secondary structure prediction. *BMC Bioinformatics*, **5**, 105.

Durbin,R. *et al*. (1999) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge.

Giegerich,R. *et al*. (2004a) A discipline of dynamic programming over sequence data. *Sci. Comp. Program*., **51**, 215–263.

Giegerich,R. *et al*. (2004b) Abstract Shapes of RNA. *Nucleic Acids Res*., **32**, 4843–4851.

Griffiths-Jones,S. *et al*. (2005) Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res*., **33**, D121–D124.

Hamada,M. *et al*. (2009) Prediction of RNA secondary structure using generalized centroid estimators. *Bioinformatics*, **25**, 465–473.

Havgaard,J.H. *et al*. (2007) Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Comput. Biol*., **3**, 1896–1908.

Hofacker,I.L. *et al*. (1994) Fast folding and comparison of RNA secondary structures. *Monatsh. Chem*., **125**, 167–188.

Janssen,S. *et al*. (2008) Shape based indexing for faster search of RNA family databases. *BMC Bioinformatics*, **9**, 131.

Lorenz,W.A. *et al*. (2008) Asymptotics of RNA shapes. *J. Comput. Biol*., **15**, 31–63.

Lu,J. *et al*. (2008) The birth and death of microRNA genes in Drosophila. *Nat. Genet*., **40**, 351–355.

Mandal,M. and Breaker,R.R. (2004) Gene regulation by riboswitches. *Nat. Rev. Mol. Cell Biol*., **5**, 451–463.

Mathews,D.H. *et al*. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol*., **288**, 911–940.

Mathews,D.H. and Turner,D.H. (2006) Prediction of RNA secondary structure by free energy minimization. *Curr. Opin. Struct. Biol*., **16**, 270–278.

McCaskill,J.S. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.

Meyer,I.M. and Miklós,I. (2004) Co-transcriptional folding is encoded within RNA genes. *BMC Mol. Biol*., **5**, 10.

Nebel,M.E. and Scheid,A. (2009) On quantitative effects of RNA shape abstraction. *Theory Biosci*., **128**, 211–225.

Reeder,J. *et al*. (2007) Locomotif: from graphical motif description to RNA motif search. *Bioinformatics*, **23**, i392.

Reeder,J. and Giegerich,R. (2005) Consensus shapes: an alternative to the Sankoff algorithm for RNA consensus structure prediction. *Bioinformatics*, **21**, 3516–3523.

Steffen,P. *et al*. (2006) RNAshapes: an integrated RNA analysis package based on abstract shapes. *Bioinformatics*, **22**, 500–503.

Voß,B. *et al*. (2006) Complete probabilistic analysis of RNA shapes. *BMC Biol*., **4**, 5.

Waldminghaus,T. *et al*. (2009) The Escherichia coli ibpA thermometer is comprised of stable and unstable structural elements. *RNA Biol*., **6**. Available at http://www.ncbi.nlm.nih.gov/pubmed/19535917 [Epub ahead of print, September 14, 2009].

Wuchty,S. *et al*. (1999) Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, **49**, 145–165.