Journal of Bioinformatics and Computational Biology
Vol. 10, No. 2 (2012) 1241007 (16 pages)

DOI: 10.1142/S0219720012410077

Imperial College Press
www.icpress.co.uk

PRACTICALITY AND TIME COMPLEXITY OF A SPARSIFIED RNA FOLDING ALGORITHM

SLAVICA DIMITRIEVA* and PHILIPP BUCHER†

Swiss Institute of Bioinformatics
and Swiss Institute for Experimental Cancer Research (ISREC)
Swiss Federal Institute of Technology (EPFL)
Lausanne, 1015, Switzerland
**slavica.dimitrieva@epfl.ch*
†*philipp.bucher@epfl.ch*

Received 10 September 2011
Revised 27 December 2011
Accepted 8 January 2012
Published 14 March 2012

Commonly used RNA folding programs compute the minimum free energy structure of a sequence under the pseudoknot exclusion constraint. They are based on Zuker's algorithm which runs in time $O(n^3)$. Recently, it has been claimed that RNA folding can be achieved in average time $O(n^2)$ using a sparsification technique. A proof of quadratic time complexity was based on the assumption that computational RNA folding obeys the "polymer-zeta property". Several variants of sparse RNA folding algorithms were later developed. Here, we present our own version, which is readily applicable to existing RNA folding programs, as it is extremely simple and does not require any new data structure. We applied it to the widely used Vienna RNAfold program, to create sibRNAfold, the first public sparsified version of a standard RNA folding program. To gain a better understanding of the time complexity of sparsified RNA folding in general, we carried out a thorough run time analysis with synthetic random sequences, both in the context of energy minimization and base pairing maximization. Contrary to previous claims, the asymptotic time complexity of a sparsified RNA folding algorithm using standard energy parameters remains $O(n^3)$ under a wide variety of conditions. Consistent with our run-time analysis, we found that RNA folding does not obey the "polymer-zeta property" as claimed previously. Yet, a basic version of a sparsified RNA folding algorithm provides 15- to 50-fold speed gain. Surprisingly, the same sparsification technique has a different effect when applied to base pairing optimization. There, its asymptotic running time complexity appears to be either quadratic or cubic depending on the base composition. The code used in this work is available at: http://sibRNAfold.sourceforge.net/.

Keywords: RNA folding; polymer-zeta property; sparsification.

1. Introduction

An RNA molecule is a single-stranded polymer that folds upon itself by forming base pairs. The ensemble of paired bases is called *RNA secondary structure* or *RNA folding*. As the biological function of an RNA depends on its secondary structure,

RNA folding programs have many applications and are widely used in RNA research. An RNA secondary structure can be described as an ensemble of different loops (hairpins, interior loops, multi-branch loops).[1,2] Assuming that the folding does not contain crossing base pairs (pseudoknots), dynamic programming algorithms using various scoring functions have been devised for computational prediction of the optimal structure.[2,3] These algorithms, which either maximize the number of base-pairs or minimize the free energy of the secondary structure, have time complexity $O(n^3)$, where n is the length of the sequence. Several computational speed-ups have been proposed in the last decades. In a theoretical paper,[4] Akutsu suggested a rather complex algorithm with time-bound $O(n^3(\log\log n)^{1/2}/(\log n)^{1/2})$. Assuming a discrete scoring scheme, Frid and Gusfield[5] applied the so-called "Four-Russians" technique reducing the time complexity to $O(n^3/\log n)$. However, to our knowledge, these approaches have never been implemented in an RNA folding program. Sparsification techniques for reducing the time and space complexity have also been applied to the problem of RNA folding.[6–9] Eppstein *et al.* proposed a sparsified dynamic programming algorithm for speeding-up the interior loop calculations, when the optimal structure contains no multi-branch loops.[7] Recently, by exploiting the triangular inequality property, sparsification was applied on the multi-loop computations.[8] In this approach, the multi-loop computations are executed conditionally and the optimal closed substructures are kept in a so called "candidate list" for later use. This work claims that computational RNA folding obeys the "polymer-zeta property",[8] which states that the probability that a base forms a pair with another base that is m monomers apart is $b \cdot m^{-c}$, where b and c are constants such that $c > 1$ and $b = 1$. Consequently, the study concludes that RNA folding using a sparsified version of the minimum free energy (MFE) algorithm can be achieved in time $O(n^2)$ on average.[8] However, this conclusion was called into question by computer simulations by another group.[10] Later, Backofen *et al.* showed how the space complexity for the base-pairing maximization variant of the RNA folding problem could also be improved using sparsification.[9] The time and space complexity of such an algorithm were expressed in terms of a sparsity parameter Z that satisfies $n \leq Z \leq n^2$ yielding bounds for time and space of $O(nZ)$ and $O(Z)$, respectively. Sparsification techniques were further applied to the problem of RNA simultaneous alignment with folding[9,11] and to the problem of RNA–RNA interaction prediction.[12] The analyses in the last two papers were based on the assumption that the "polymer-zeta property" with $c > 1$ holds for each of the RNA sequences under consideration. Recently sparsification was also applied to algorithms for prediction of pseudoknotted RNA structures.[13]

In this work, we introduce another version of sparsification for RNA folding algorithms and perform a thorough analysis of the time complexity of sparsified RNA folding algorithms in general. Although discovered independently, our version is closely related to the ones introduced in Refs. 8 and 9. However, in our version, the speed-gain is achieved solely by re-ordering and conditional execution of elementary arithmetic operations without requiring any additional data structure,

which makes the implementation of our approach extremely simple. To prove practicality, we implemented our sparsification approach by modifying the code of the widely used *RNAfold* and *RNAalifold* programs from the *Vienna RNA* package. Next, we carried out a thorough run time analysis with real and synthetic RNA sequences, both in the context of energy minimization and base-pairing maximization. To gain a better understanding of the time complexity of sparsified RNA folding, we varied the base composition, the folding temperature and multi-loop parameters of the energy function. Our analyses contradict the previous claims that computational RNA folding obeys the polymer-zeta property with $c > 1$. The time complexity of a sparsified RNA folding algorithm with real energy parameters remains cubic or near cubic under a wide variety of conditions. In contrast, we observe quadratic time complexity of a sparsified base-pairing maximization algorithm for sequences with a skewed base composition (high A + C content).

2. Preliminaries

Minimum free energy RNA folding programs use energy functions that take into account stabilizing stacking energies for neighboring base-pairs in double-helical regions and destabilizing energies for loops containing unpaired bases. Some jargon, taken from Ref. 14, will be introduced to explain the basic form of the energy function. A secondary structure is represented as a list of non-crossing base pairs specified by indices to sequence positions. Each base in the sequence can be part of at most one base-pair. By convention, we order pairs by increasing position numbers. Two relations between base-pairs will be introduced: (i) k, l is said to be "interior" to a base pair i, j if $i < k < l < j$, (ii) k, l is said to be "immediately interior" to i, j if there is no base pair p, q such that $i < p < k < l < q < j$.

Central to the energy function is the concept of a "loop" since the total energy of a structure can be expressed as the sum over all loop energies. Each base pair in a structure closes exactly one loop. A loop closed by i, j comprises i, j itself, all base-pairs which are immediately interior to it and the unpaired regions between these base-pairs. The mathematical formula used to compute the energy for a loop depends on the loop type. Loops with zero interior base-pairs are called hairpin loops. Loops with exactly one immediately interior base-pair k, l are called "interior"; they comprise three subtypes:

$k = i + 1$ and $l = j - 1$	: stacked base pairs (no unpaired regions)
$k > i + 1$ and $l < j - 1$	: true interior loop (unpaired regions on both sides)
all other cases	: bulge loop (unpaired bases on one side only)

Loops with more than one immediately interior base pair are called multi-branched or multi-loops. Note further that a secondary structure may also contain unpaired sequence regions that are not part of any loop. Those are sometimes treated as a special loop closed by the unpaired ends of the structure.

The energies for stacked base pairs are tabulated in a 6×6 table with columns and rows corresponding to the six canonical base pairs found in RNA. These tables explicitly list the entropic and enthalpic components for the stacking energies, which allows for temperature adjustment of the global energy function. The energies of all other loop types by default depend only on the loop type and the length of the unpaired sequence segments in the loop. However, the extended thermodynamic parameter sets used by current programs[14] include large tables of sequence-specific loop energies for short loops. In addition, they contain negative energy values for stacking interactions between a terminal base-pair of a helical region and adjacent unpaired bases. Such interactions are referred to as $5'$ and $3'$ dangles.

A basic version of an RNA folding algorithm, which minimizes an energy function of the above type, uses the following recursion:

$$W(i,j) = \min \begin{cases} V(i,j) \\ W(i+1,j) \\ W(i,j-1) \\ \min\limits_{i \le k < j} \{W(i,k) + W(k+1,j)\} \end{cases}$$

$$V(i,j) = \min \begin{cases} \text{Hairpin}(i,j) \\ \min\limits_{i<k<l<j} \text{Interior}(i,j,k,l) + V(k,l) \\ \text{Multi}(i,j) + W(i+1,j-1) \end{cases}$$

Two values are computed for each subsequence i to j. V_{ij} is the MFE under the constraint that bases i and j form a base-pair; W_{ij} is the globally MFE. "Hairpin", "Interior", and "Multi" are functions that return the energy values of the corresponding loops. Note that "Interior" is a heterogeneous function returning energy values for stacking base pairs, bulge loops, and true interior loops, all based on different parameter sets. This basic version is compatible with a multi-loop scoring function that returns a constant positive value a if i, j correspond to a canonical RNA base pair and infinity otherwise. Current RNA folding programs use more elaborate mutliloop scoring functions that require computation of additional auxiliary arrays. Note further that dangling interactions are not explicitly treated by this recursion.

The time complexity of the above algorithm is in principle $O(n^4)$ due to the loop over all possible interior loops. However, it is common to restrict the evaluation of this term to loops of a maximal length e.g. $k - i + j - l < 30$. This renders the total time spent on interior loops quadratic. Alternatively, near quadratic solutions for interior loop evaluation have been described and could be used instead.[15] Therefore, the rate-limiting step which has time complexity $O(n^3)$ is the loop over all possible ways to join two optimal substructures:

$$\min_{i \le k < j} \{W(i,k) + W(k+1,j)\}$$

There are a number of variations to classical MFE folding, most notably probabilistic and covariance-guided folding. Probabilistic folding takes into account the fact that RNA molecules exist as an ensemble of energetically similar structures. The McCaskill algorithm converts energy differences into probabilities according to the Boltzmann equation and computes the sum of the probabilities of all structures containing a particular base pair.[16] The resulting pair probabilities can be displayed as a dot matrix to assess the confidence of predicted structural elements. Covariance-guided RNA folding exploits information contained in a multiple sequence alignment and is based on the assumption that pairs of interacting bases vary in a coordinated fashion such as to preserve base complementarity. Covariance is expressed as a pseudo-energy that can be added to a thermodynamic energy function adjusted to multiple alignments.[17] Adaptation of the basic RNA folding algorithm to covariance-guided folding is then straightforward. Additional variations of the standard secondary structure prediction problem include: (i) the folding of circular RNAs, (ii) the simultaneous co-folding of two interacting RNA molecules, and (iii) the search for locally stable substructures in long sequences. All these functions are implemented in version 1.8.4 of the Vienna RNA secondary structure package (http://www.tbi.univie.ac.at/RNA/), which we used as a software platform to benchmark our algorithmic improvements.

On another note, classical RNA folding algorithms subject to the pseudo-knot exclusion constraint, face today competition by heuristic algorithms allowing for pseudo-knots, see for instance Refs. 18 and 19. An introduction of these algorithms is, however, beyond the scope of this paper.

3. Sparsified RNA Folding

3.1. *Our sparsified version of the Nussinov algorithm*

The speed-up principle based on sparsification will first be explained in the context of the Nussinov algorithm,[3] which maximizes the number of non-intersecting canonical base pairs that can be formed by an RNA sequence. After adequate initializations, this number can be obtained by the following recursion:

$$F(i,j) = \max\{F(i+1,j-1) + \delta(i,j), \quad \max_{i \le k < j}\{F(i,k) + F(k+1,j)\}\}$$

Here $F(i,j)$ is the best score (maximal number of bases pairs) for the subsequence i to j. The function $\delta(i,j)$ returns 1 if the bases at positions i and j can form a canonical base pair, and 0 otherwise.

Note that the optimal structure for a subsequence i to j is obtained either by a loop-closing operation (first term) or by joining two optimal substructures (second term). The latter requires evaluation of all possible ways to split a subsequence into two. This is the speed-limiting step leading to time complexity $O(n^3)$.

Computation of the $F(i,j)$ values for all subsequences is carried out during the so-called fill-stage of the Nussinov algorithm. Once all these values are known, the

optimal secondary structure is obtained by a fast trace-back procedure. The algorithmic modifications described here concern only the fill-stage and the trace-back procedure need not to be changed. A complete version of the fill-stage of the non-sparsified algorithm is given below:

$$\begin{aligned}
\text{Initialization} &: F(1,1)=0;\ \text{for } i=2 \text{ to } N:\{F(i,i-1)=0;\ F(i,i)=0\}\\
\text{Iteration} &: \text{for } i=N-1 \text{ to } 1:\\
&\quad \text{for } j=i+1 \text{ to } N:\\
&\qquad F(i,j)=F(i+1,j-1)+\delta(i,j)\\
&\qquad \text{for } k=i \text{ to } j-1:\{F(i,j)=\max\{F(i,j),F(i,k)+F(k+1,j)\}\}
\end{aligned}$$

In this version, the subsequence score matrix F is computed by filling rows from left to right starting at the bottom. The elements of the matrix F could be computed in a different order, see for instance the presentation in Ref. 21. What is more important in this context is that the elementary computations can be carried out with a different loop-architecture:

$$\begin{aligned}
\text{Initialization} &: \text{for } i=1 \text{ to } N:\{F(1,i)=0\}\\
&\quad \text{for } i=2 \text{ to } N:\{\text{for } j=i-1 \text{ to } N:\{F(i,j)=0\}\}\\
\text{Iteration} &: \text{for } i=N-1 \text{ to } 1:\\
&\quad \text{for } j=i+1 \text{ to } N:\\
&\qquad F(i,j)=\max\{F(i,j),F(i+1,j),F(i+1,j-1)+\delta(i,j)\}\\
&\qquad \text{for } k=j+1 \text{ to } N:\{F(i,k)=\max\{F(i,k),F(i,j)+F(j+1,k)\}\}
\end{aligned}$$

Here, the optimal scores for composite structures for a subsequence i to k are computed "in advance" i.e. before the matrix element $F(i,k)$ is reached by the two outer loops. In fact, immediately after the final value $F(i,j)$ has been determined, the innermost loop already evaluates the scores of the secondary structures that could be obtained by joining the best structure of subsequence i to j with the best structure of adjacent subsequence $j+1$ to k. Later, when the algorithm computes the final value of $F(i,k)$, all composite structure scores for subsequence i to k have already been evaluated. It is obvious that this new loop configuration leads to the same matrix F as the original algorithm.

The purpose of this reordering of elementary arithmetic operations is to allow for conditional execution of the innermost loop. The innermost loop needs be carried out only if the best score for subsequence i to j can only be obtained by loop-closing. Otherwise, if $F(i,j)$ is equal to the score of a composite structure, there must be an index l satisfying $i \le l < j$ such that

$$F(i,j)=F(i,l)+F(l+1,j).$$

We further note that for any index k, $j<k\le N$:

$$F(i,j)+F(j+1,k)=F(i,l)+F(l+1,j)+F(j+1,k)\le F(i,l)+F(l+1,k).$$

However, the term $F(i,l)+F(l+1,k)$ was already evaluated when the two outer loops reached the matrix element $F(i,l)$. Therefore, computation of $F(i,j)+F(j+1,k)$

is obsolete as the resulting value cannot be greater than the current value of $F(i,k)$. A complete version of the modified Nussinov algorithm with conditional innermost loop execution is shown below:

$$
\begin{aligned}
\text{Initialization} &: \text{for } i = 1 \text{ to } N : \{F(1,i) = 0\} \\
&\quad \text{for } i = 2 \text{ to } N : \{\text{for } j = i-1 \text{ to } N : \{F(i,j) = 0\}\} \\
\text{Iteration} &: \text{for } i = N-1 \text{ to } 1 : \\
&\quad \text{for } j = i+1 \text{ to } N : \\
&\qquad F(i,j) = \max\{F(i,j), F(i+1,j)\} \\
&\qquad \text{if } F(i+1,j-1) + \delta(i,j) > F(i,j) : \\
&\qquad\quad F(i,j) = F(i+1,j-1) + \delta(i,j) \\
&\qquad\quad \text{for } k = j+1 \text{ to } N : \{F(i,k) = \max\{F(i,k), F(i,j) + F(j+1,k)\}\}
\end{aligned}
$$

3.2. *Our sparsified version of standard RNA folding algorithm*

The algorithmic trick described above is readily applicable to standard RNA folding algorithms that minimize a realistic energy function under the pseudo-knot exclusion constraint. In this work, we have chosen to use the *Vienna RNAfold* program as a realistic test platform. Consequently, we describe the modifications with regard to the specific algorithm implemented in this program.

The algorithm used by the *RNAfold* code is shown in Fig. 1 along with our modifications allowing for speed-up. We emphasize that this pseudo-code reflects the

$$
\begin{aligned}
&\text{for } i = N \text{ to } 1 : \\
&\quad \text{for } j = i \text{ to } N : \\
&\qquad C(i,j) = \min\ \{\text{Hairpin}(i,j),\ \min_{i<k<l<j} \text{Interior}(i,j,k,l) + C(k,l)\ \} \\
&\qquad \text{if } i,j \text{ can be paired} : C(i,j) = \min\ \{C(i,j), M_2(i+1,j-1) + a\ \} \\
&\qquad \text{original} : \begin{cases} M(i,j) = \min\ \{M(i+1,j)+c, M(i,j-1)+c, C(i,j)+b\} \\ \text{for } k = i \text{ to } j-1 : \\ \quad M_2(i,j) = \min\ \{M(i,k) + M(k+1,j),\ M_2(i,j)\} \end{cases} \\
&\qquad \text{modified} : \begin{cases} M(i,j) = \min\ \{M(i+1,j)+c, M(i,j-1)+c\} \\ \text{if } C(i,j)+b < M(i,j)\ : \\ \quad M(i,j) = C(i,j)+b \\ \quad \text{for } k = j+1 \text{ to } N : \\ \quad\quad M_2(i,k) = \min\ \{M(i,j) + M(j+1,k),\ M_2(i,k)\} \end{cases} \\
&\qquad M(i,j) = \min\ \{M(i,j), M_2(i,j)\ \} \\
&F(1,1) = 0 \\
&\text{for } i = 2 \text{ to } N\ : \\
&\quad F(1,i) = \min\ \{F(1,i-1), C(1,i),\ \min_{i<k<j} \{F(1,k-1) + C(k,i)\}\ \}
\end{aligned}
$$

Fig. 1. Pseudo-code for the original and modified RNAfold algorithms. The code corresponds to the algorithm used by RNAfold with option $-d0$ (no energy benefits for dangling bases). The parts that differ between the two versions are labeled "original" and "modified."

source code used in this study, and we are aware of the fact that it differs from earlier descriptions of the folding algorithm implemented in the *Vienna* package.[20,22]

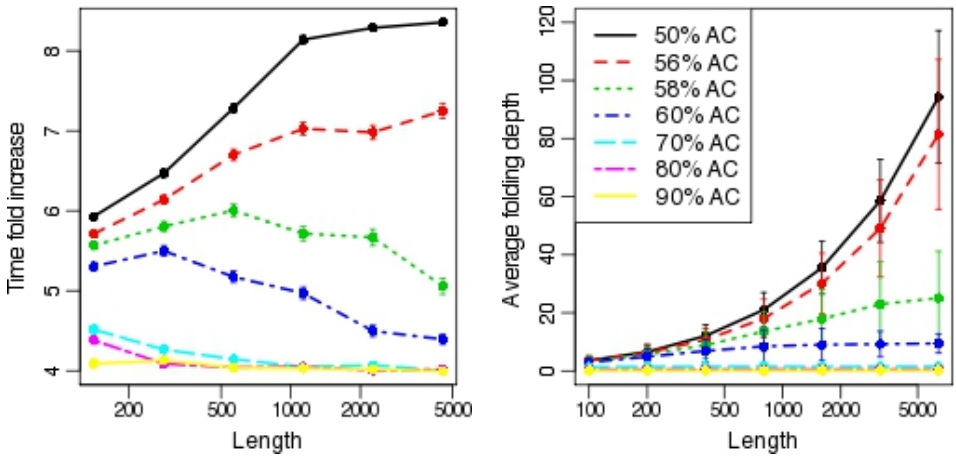
The algorithm shown in Fig. 1 computes two complete arrays: $C(i, j)$ is the MFE of a closed RNA structure of subsequence i to j, $M(i, j)$ is the MFE of subsequence i to j scored as a multi-loop component. The array F contains the MFE for all subsequences starting at position 1. $M_2(i, j)$ is the MFE of subsequence i to j scored as the interior part of a multi-loop, and conditional on the presence of at least two interior base-pairs (otherwise it will have an infinitely high value). It needs be kept in memory only temporarily. The computation of M_2 makes sure that the multi-loop scoring system will be applied to multi-loops only. The pseudo-code shown does not take into account $5'$ and $3'$ dangles. This corresponds to the option $-d0$ of the program *RNAfold*. The same program offers three additional dangling options. With $-d1$, an unpaired base can participate in at most one dangling interaction; with option $-d2$, this check is ignored; the $-d3$ option allows for coaxial stacking of helices in a multi-loop. The algorithmic treatment of dangles is described in Ref. 14. The modification described here is compatible with dangling options 0, 1 and 2, and is implemented for all of these variants.

4. Results

4.1. *Empirical time complexity analysis of the sparsified Nussinov algorithm*

The speed gain obtained with the sparsified algorithms obviously depends on the frequency by which the condition requiring innermost loop execution is fulfilled. This frequency in turn may depend on the base composition of the sequences to be folded. Therefore, we carried out tests with sequence sets of increasing length and increasing content of A + C. When varying the A + C content, we kept the frequencies of A equal to the frequencies of C, and likewise the frequencies of G equal to the frequencies of U. The frequencies of A and C were changed because these two bases have only one canonical interaction partner and thus are expected to favor less folded structures.

The results of the running time analysis are shown in Fig. 2(a). The graph shows the fold-increase in time resulting from doubling the sequence length. For long sequences, this value should approach 8 for cubic time complexity and 4 for quadratic. For an unbiased base composition with 50% A + C, we clearly observe cubic time complexity. For sequences with an A + C content of 60% or higher, we see a trend toward quadratic time complexity. The curves for 56% and 58% A + C are less clear but seem to asymptotically approach cubic and quadratic complexity, respectively. Taken together, our results suggest that the sparsified Nussinov algorithm has quadratic time complexity only for sequences with biased base composition. A sharp transition from cubic to quadratic time complexity occurs near 57% A + C. This transition is very reminiscent of the phase transition behavior of local alignment scoring systems described in Ref. 23.



(a) (b)

Fig. 2. Sequence base composition dependence of the speed gain and folding depth using the modified Nussinov algorithm. The modified Nussinov algorithm was applied to sets of 1000 sequences of length $100, 200 \ldots, 3200$ with varying base composition. (a) The fold-increase in the number of elementary operations resulting from doubling the sequence length is plotted against the geometric mean of the shorter and longer sequence length. The error bars represent the standard error of the ratios calculated from the standard deviations and means of the number of elementary operations (see formula (1) in the supplementary material for details) (b) Average folding depth of the optimal structures as defined in the main text. The error bars represent the standard deviation of the folding depth.

We were wondering whether the time complexity correlates with topological properties of the optimal structures returned by the Nussinov algorithm. Specifically, we were looking at the average folding depth of the structures. The folding depth of an individual base in a structure is defined as the number of nested base pairs enclosing it. Figure 2(b) shows the average folding depth for different sequence sets. As a general trend, we observe a sequence length−independent constant folding depth for base compositions which lead to quadratic time complexity and monotonous increase in the cubic case. This result can easily be rationalized if we assume that the length invariant behavior of the average folding depth reflects concatemeric structures composed of closed substructures not exceeding a certain limit size. In this case, the optimal structures for subsequences exceeding the limit size are never closed and hence will not trigger innermost loop execution in the sparsified Nussinov algorithm.

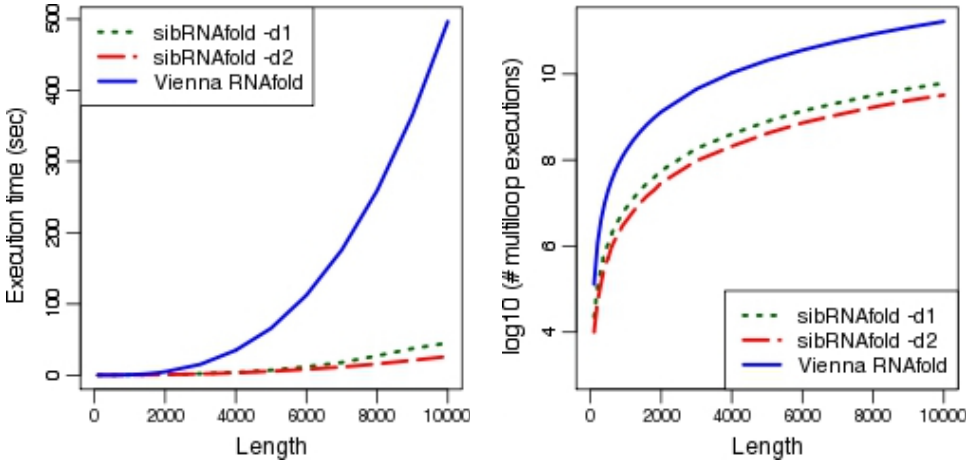
4.2. *Empirical time complexity analysis of the sparsified RNAfold algorithm*

We applied the same speed-up strategy to the *RNAfold* program from the *Vienna* package and created the program *sibRNAfold*, the first publicly available sparsified RNA folding program that uses standard energy parameters. The implementation of our sparsification version turned out to be surprisingly simple. Only a few lines in

the function *fold* had to be changed in order to implement conditional innermost loop execution. Other modifications were necessary to allow processing of sequences longer than 32 kb. *sibRNAfold* produces identical result as *Vienna RNAfold* but is more than an order of magnitude faster. It thereby extends the application range of RNA folding to longer RNA sequences up to 100 kb. Since the input and output routines were not changed, *sibRNAfold* is compatible with existing sequence analysis pipelines using *Vienna RNAfold*.

To analyze the speed-gain of *sibRNAfold* versus *Vienna RNAfold*, we folded real and random sequences of varying length (Fig. 3). In all tests we verified that the results returned by both programs were identical. The savings in execution time ranged from a factor of 15 to 50, with higher benefits recorded for longer sequences.

For instance, the time for folding the HIV genome (9181 bp) went down from 344 s to 19 s. The SARS genomic RNA (29,751 bp) was folded in 6 min as compared to 3 h required for the original version. With *sibRNAfold* we were even able to fold the human titin mRNA (101,674 bp) in 3 h, while the estimated time with the original version is several days. We note that folding such a large mRNA may not result in a biologically relevant structure for reasons of insufficient precision of the energy parameters or kinetic inaccessibility of the MFE structure. For such long sequences, it is more reasonable to apply covariance-guided folding, where this sparsification principle is also applicable. Nevertheless, the ability to fold such long sequences in reasonable time is interesting. Note that for a better estimation of the contribution of the cubic step, we restricted the maximum length of an interior loop to 10 instead of 30, the default value used by *RNAfold*.



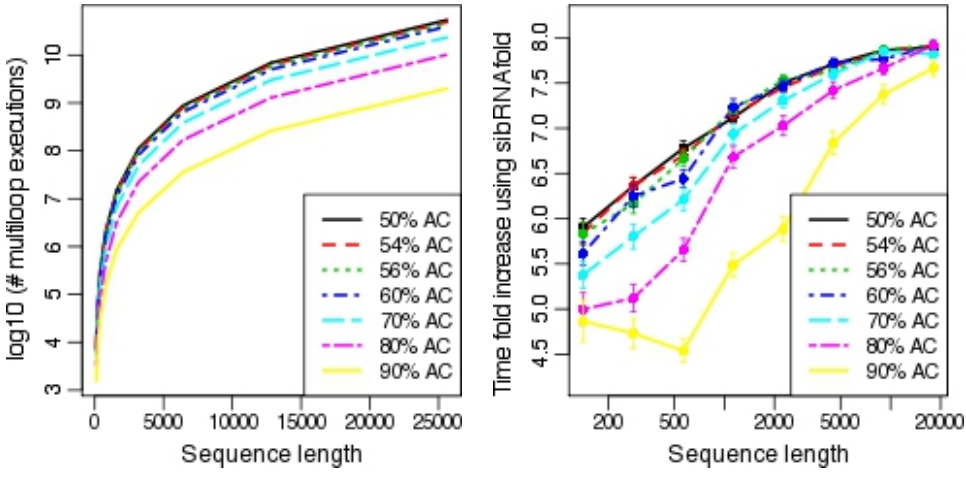
(a) (b)

Fig. 3. Speed comparison of sibRNAfold versus the Vienna RNAfold program. The values represent the average over 10 random sequences of the same length. (a) Runtime in s. The standard deviation of all plotted values is < 2 s and therefore too small to be shown; (b) Number of join operations on a logarithmic vertical scale. The standard deviations of all plotted values is < 0.05 on a log scale.

Sparsification is applicable to multiple alignment−based folding. We applied the same speed-up strategy to the *RNAalifold* program from the *Vienna* package, by changing only a few lines in the function *alifold* and we observed a speed gain in the same range (Fig. S1).

To gain an estimate of the time complexity of the sparsified version versus the original one, we calculated the time fold increase resulting from doubling the sequence length. As expected, with synthetic sequences of uniform base composition (50% A + C) we observed cubic time complexity. Increasing the A + C content resulted in a speed gain. However, in contrast to the results obtained with the sparsified Nussinov algorithm, the time complexity remains cubic independently of the sequence base composition (Fig. 4).

Next we addressed the questions whether the speed gain depends on the energy parameters. *RNAfold* allows for automatic temperature adjustment of the energy function via a command line option. We thus computed the secondary structures of random RNA sequences of 50% A + C content at different temperatures (Fig. S2). We note that the number of join operations required for MFE computations decreases with increasing temperature. However, the time complexity again remains cubic at least up to 70°C.



(a) (b)

Fig. 4. Dependence of the number of join operations and time-fold increase on sequence composition (computed with sibRNAfold $-d2$ option). The mean values and ratios are estimated from sets of 100 random sequences of the same length. (a) Number of join operations on a logarithmic vertical scale. The standard deviation of all plotted values is < 0.1 and therefore too small to be shown; (b) Fold increase in the number of elementary operations resulting from doubling the sequence length plotted against the geometric mean of the shorter and longer sequence length. The error bars represent the standard error of the ratios calculated from the standard deviations and means of the number of elementary operations (see formula (1) in the supplementary material for details).

Since the rate-limiting cubic step concerns multi-loops only, we analyzed in more details the effect of multi-loop parameters on run time (Table S1, Fig. S3). The program *RNAfold* scores multi-loops as follows:

$$\Delta G_{\text{loop}} = a + bh + cn$$

Here, h is the number of immediately interior base-pairs, n is the number of unpaired bases in the loop, and a, b, c are the parameters of the energy functions. The following observations can be made: among the parameter sets that predict structures with similar free energy, the one with a non-zero value for c leads to considerably higher execution time. However, the running time of the algorithm stays cubic independently of the values for the multi-loop parameters.

4.3. *The Polymer Zeta Property of RNA folding*

The polymer-zeta property makes a statement about the probability that the terminal bases of a folded RNA are paired. It implies that this probability exponentially decreases to zero with increasing sequence length. More precisely, it approaches $b \cdot m^{-c}$, where m is the length of the sequence and b and c are constants such that $c > 1$ and $b = 1$. In a previous work by Wexler *et al.*,[8] the exponent c was empirically estimated to be 1.47 for optimal RNA secondary structures according to the energy model of Zuker and Steigler.[2] Theoretical analysis presented in the same paper suggested that a sparsified RNA folding program would indeed have quadratic time complexity if $c > 1$, and $O(n^2 \cdot n^{1-c}(\log n)^c)$ if $c < 1$.

We challenged the hypothesis that computational RNA folding obeys the polymer-zeta property with an exponent greater than one by applying the sparsified Nussinov and RNAfold algorithms to sets of random RNA sequences with varying base composition and length, and subsequently counting the fraction of closed structures. The results of this analysis are presented in Fig. 5 and Table 1. For the cases where we observed quadratic time complexity, we observe a monotonous decrease in the fraction of closed structure with increasing sequence length, which is compatible with the polymer-zeta property. For all other cases, the fraction of closed structures appears to stabilize at some positive value, which would imply cubic time complexity. We tried to interpret the curves presented in Fig. 5 in the light of the polymer zeta function by non-linear least-square fitting (Table 1). Again, this analysis yielded results compatible with our running time analysis. The estimates for the exponent c are greater than one only for those cases where we observed quadratic time complexity. For standard RNA folding and uniform base composition, the estimated c is very close to zero. In summary, our time complexity analysis contradicts the claim that computational RNA folding with standard energy functions obeys the polymer-zeta property with parameter $c > 1$. However, if we look carefully at the benchmark results from Ref. 9, we see that our results are in accordance with the ones published there. In Fig. 12 of Ref. 9, Backofen *et al.*

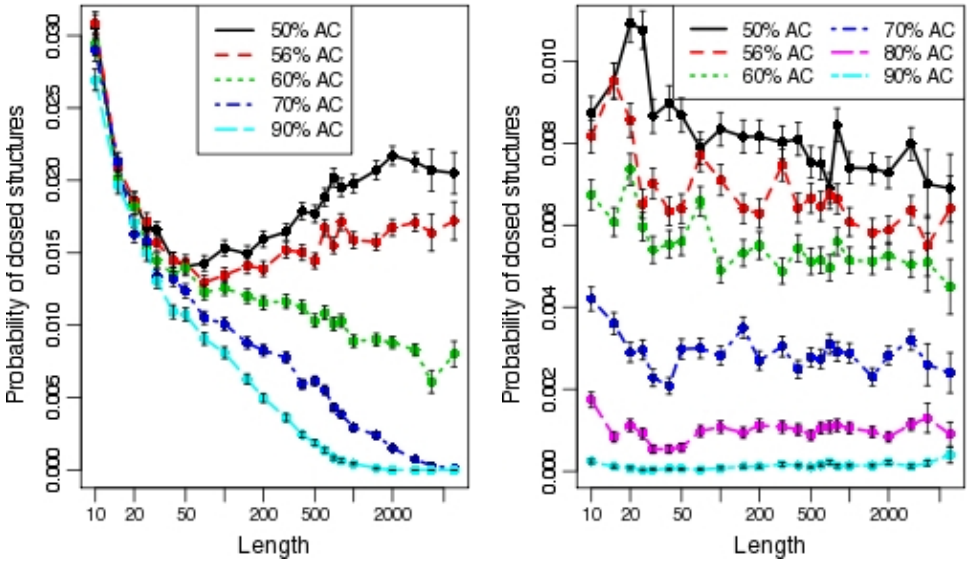


Fig. 5. Probability that the optimal folding corresponds to a closed structure as a function of the sequence size. The folding algorithms were applied to sets of 50,000 sequences with different lengths and with varying base composition. (a) Results using the modified Nussinov algorithm; (b) Results using sibRNAfold (modified Vienna RNAfold). The standard deviation of the plotted values is calculated as $\sqrt{k}/N$, where k is the number of occurrences of closed structures and N is the number of sequences tested.

Table 1. Estimates for the constants b and c from the equation describing the polymer zeta property for RNA sequences.

	Modified Nussinov algorithm		sibRNAfold algorithm	
AC content (in %)	b	c	b	c
50	0.017	−0.018	0.011	0.055
56	0.033	0.195	0.009	0.051
58	0.072	0.445	0.008	0.055
60	0.109	0.615	0.007	0.047
70	0.428	1.311	0.003	0.031
80	1.377	2.129	0.001	−0.015
90	3.881	3.199	3.1e-05	−0.240

Note: The estimates are based on folding of sets of 50,000 sequences with different AC content using the modified Nussinov algorithm and sibRNAfold. They are computed by fitting a nonlinear least-squares model using the R statistical analysis package, www.r-project.org, (*nls* function).

plotted the percentage of multi-loop executions of the sparsified algorithm (relative to the number of multi-loop executions required by the original algorithm) as a function of the sequence length. Clearly, for long sequences, this value is stabilizing at 2% rather than asymptotically approaching zero.[9]

5. Discussion and Conclusions

We have presented our own sparsified version of classical RNA folding algorithms. We have implemented it by modifying the widely used *Vienna RNAfold* program, to create *sibRNAfold*, a publicly available sparsified RNA folding program using standard energy parameters. We deliberately opted for modification of existing code rather than reprogramming to ensure identical results with a trusted implementation. By an extensive empirical analysis, we have shown that the time complexity of a sparsified RNA folding algorithm based on energy minimization remains cubic or near cubic, independently of the energy function used and the base composition of the RNA sequence. Yet, a basic version of sparsification requiring minimal changes of existing code provides up to 50-fold speed gain for RNA folding.

Our speed analysis carried out with the modified Nussinov algorithm reveals a conspicuous phase-transition behavior dependent on the base composition, which is reflected by both time complexity and folding depth of the optimal structures. This is a novel finding, perhaps biologically irrelevant but highly interesting from an algorithmic viewpoint. Note that Backofen *et al.*[9] already noticed that the speed-gain of a sparsified base-pairing maximization algorithm varies as a function of the base composition. However, they analyzed this effect only for sequences of length 500 and therefore could not make any inference regarding the impact of the base-composition on the asymptotic time complexity.

We conclude with a brief comparison of our work with other recently published improvements of the basic RNA folding algorithm. In doing so, we will challenge previous claims that RNA folding with a standard energy function can be achieved in near-quadratic time.

Sparsification strategies based on the same principle like the one described here were previously described in Refs. 8 and 9. In fact, we use the same condition to restrict the execution of structure joining operations as in Ref. 8, and therefore the time complexity should stay the same. In contrast to our approach, in Refs. 8 and 9, the optimal closed substructures are kept in a candidate list for later use. This has the principle advantage that the memory requirements are reduced as well.[9] On the other hand, our approach is easier to apply to existing code, as it does not require introduction of a new data structure and modification of the trace-back procedure.

Sparsification was implemented in a program called *CanididateFold*,[8] which reportedly computed the optimal secondary structure according to the energy model of Zuker and Stiegler.[2] The run-time of *CandidateFold* was reported to be quadratic in sequence length, a speed-gain that we were not able to achieve with our approach. We could not further investigate the reasons for the discrepancy in time complexity since *CandidateFold* is no longer maintained.[24] However, we suspect that the reason for this is that the reported results using *CandidateFold* were for sequences with short length (up to 1000 bp). For such sequences, the running time of the algorithm is more realistically approximated as $\alpha \cdot n^3 + \beta \cdot n^2 + o(n^2)$, where α and β are constants and n is the sequence length. Assuming that n is small and $\beta \gg \alpha$, the

running time of the algorithm will be dominated by the quadratic term. Therefore, for n below a certain threshold, one can still observe quadratic time complexity even if the asymptotic time complexity is cubic. Our empirical results on the speed of a sparsified RNA folding algorithm are realistic in the sense that they were obtained with a program that computes exactly the same structures as a widely used RNA folding program. With the default energy parameters and with base sequences that resemble natural RNAs in terms of base composition, we obtain a significant speed gain but still cubic time complexity. On the other hand, we observed quadratic time complexity of the sparsified Nussinov algorithm only for sequences enriched in A + C. Moreover, there are indications that the modified *RNAfold* program runs in sub-cubic time with modified multi-loop scoring function for sequences with 90% A + C. Overall, our results speak against previous claims that a sparsification of an RNA folding algorithm can bring down its time complexity to near quadratic. Furthermore, our analysis contradicts the claim that computational RNA folding obeys the polymer-zeta property with parameter $c > 1$.

Acknowledgments

SD was supported by grant PDFM33-120719 from the Swiss National Science Foundation.

References

1. Tinoco I, Uhlenbeck OC, Levine MD, Estimation of secondary structure in ribonucleic acids, *Nature* **230**:362, 1971.
2. Zuker M, Stiegler P, Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, *Nucleic Acids Res* **9**:133−148, 1981.
3. Nussinov R, Jacobson AB, Fast algorithm for predicting the secondary structure of single-stranded RNA, *Proc Natl Acad Sci-Biol* **77**:6309−6313, 1980.
4. Akutsu T, Approximation and exact algorithms for RNA secondary structure prediction and recognition of stochastic context-free languages, *J Comb Optim* **3**:321−336, 1999.
5. Frid Y, Gusfield D, A simple, practical and complete $O(n(3)/\log n)$-time algorithm for RNA folding using the Four-Russians Speedup, *Algorithm Mol Biol* doi: 10.1186/1748-7188-5-13.
6. Eppstein D, Galil Z, Giancarlo R, Italiano GF, Sparse dynamic-programming. 1. Linear cost-functions, *J Acm* **39**:519−545, 1992.
7. Eppstein D, Galil Z, Giancarlo R, Italiano GF, Sparse dynamic-programming. 2. Convex and concave cost-functions, *J ACM* **39**:546−567, 1992.
8. Wexler Y, Zilberstein C, Ziv-Ukelson M, A study of accessible motifs and RNA folding complexity, *J Comput Biol* **14**:856−872, doi: 10.1089/cmb.2007.R020.
9. Backofen R, Tsur D, Zakov S, Ziv-Ukelson M, Sparse RNA folding: Time and space efficient algorithms, *J Discrete Algorithms* **9**:12−31, 2011.
10. Starykovskaya TA, Roytberg MA, Nested arc-annotated sequences and strong fragments, *Proc. 3rd Moscow Conference on Computational Molecular Biology*, 281−283, 2007.
11. Ziv-Ukelson M, Gat-Viks I, Wexler Y, Shamir RA, Faster algorithm for simultaneous alignment and folding of RNA, *J Comput Biol* **17**:1051−1065, 2010.

12. Sahinalp SC, Salari R, Mohl M, Will S, Backofen R, Time and space efficient RNA-RNA interaction prediction via sparse folding, *Proc Res Comput Mol Biol* **6044**:473−490, 2010.
13. Mohl M, Salari R, Will S, Backofen R, Sahinalp SC, Sparsification of RNA structure prediction including pseudoknots, *Algorithms Mol Biol* **5**:39, 2010.
14. Turner DH, Mathews DH, Sabina J, Zuker M, Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure, *J Mol Biol* **288**:911−940, 1999.
15. Roytberg MA, Ogurtsov AY, Shabalina SA, Kondrashov AS, Analysis of internal loops within the RNA secondary structure in almost quadratic time, *Bioinformatics* **22**:1317−1324, 2006.
16. McCaskill JS, The equilibrium partition-function and base pair binding probabilities for RNA secondary structure, *Biopolymers* **29**:1105−1119, 1990.
17. Stadler PF, Hofacker IL, Fekete M, Secondary structure prediction for aligned RNA sequences, *J Mol Biol* **319**:1059−1066, 2002.
18. Li HW, Approximation algorithm for pseudoknotted RNA structure prediction, *Cis Workshops 2007: Int Conf Computational Intelligence and Security Workshops*, 108−111, 2007.
19. Condon A, Ren JH, Rastegari B, Hoos HH, HotKnots: Heuristic prediction of RNA secondary structures including pseudoknots, *RNA* **11**:1494−1504, 2005.
20. Hofacker IL *et al.*, Fast folding and comparison of RNA secondary structures, *Monatsh Chem* **125**:167−188, 1994.
21. Durbin R, *Biological Sequence Analysis: Probalistic Models of Proteins and Nucleic Acids* (Cambridge University Press, 1998).
22. Hofacker IL, Stadler PF, Memory efficient folding algorithms for circular RNA secondary structures, *Bioinformatics* **22**:1172−1176, 2006.
23. Waterman MS, Gordon L, Arratia R, Phase-transitions in sequence matches and nucleic-acid structure, *Proc Natl Acad Sci USA* **84**:1239−1243, 1987.
24. Ziv-Ukelson M, Personal communication.

Slavica Dimitrieva received her Diploma Degree in Electrical Egineering from University "Ss. Cyril and Methodius," Macedonia in 2004, and Masters Degree in Computer Science from ETH Zurich in 2009. Since 2009, she is a Ph.D. student at EPFL working under supervision of Dr. Philipp Bucher.

Philipp Bucher was first trained as a molecular biologist at the University of Zürich, and subsequently received his Ph.D. in Computational Biology at the Weizmann Institute of Science in Israel. He then worked as a postdoctoral fellow with Sam Karlin at Stanford University before he moved in 1991 to ISREC to continue his research in comparative molecular sequence analysis. In 1995, he was promoted as a senior scientist.