Möhl *et al. Algorithms for Molecular Biology* 2010, **5**:39
http://www.almob.org/content/5/1/39

ALGORITHMS FOR MOLECULAR BIOLOGY

RESEARCH Open Access

Sparsification of RNA structure prediction including pseudoknots

Mathias Möhl[1†], Raheleh Salari[2†], Sebastian Will[1,3†], Rolf Backofen[1,4*], S Cenk Sahinalp[2*]

Abstract

Background: Although many RNA molecules contain pseudoknots, computational prediction of pseudoknotted RNA structure is still in its infancy due to high running time and space consumption implied by the dynamic programming formulations of the problem.

Results: In this paper, we introduce sparsification to significantly speedup the dynamic programming approaches for pseudoknotted RNA structure prediction, which also lower the space requirements. Although sparsification has been applied to a number of RNA-related structure prediction problems in the past few years, we provide the first application of sparsification to pseudoknotted RNA structure prediction specifically and to handling gapped fragments more generally - which has a much more complex recursive structure than other problems to which sparsification has been applied. We analyse how to sparsify four pseudoknot structure prediction algorithms, among those the most general method available (the Rivas-Eddy algorithm) and the fastest one (Reeder-Giegerich algorithm). In all algorithms the number of "candidate" substructures to be considered is reduced.

Conclusions: Our experimental results on the sparsified Reeder-Giegerich algorithm suggest a linear speedup over the unsparsified implementation.

Background

Recently discovered catalytic and regulatory RNAs [1,2] exhibit their functionality due to specific secondary and tertiary structures [3,4]. The vast majority of computational analysis of non-coding RNAs have been restricted to nested secondary structures, neglecting pseudoknots - which are "among the most prevalent RNA structures" [5]. For example, Xaya-phoummine et al. [6] estimated that up to 30% of the base pairs in G+C-rich sequences form pseudoknots.

However the general problem of pseudoknotted RNA structure prediction is NP-hard. As a result, a number of approaches have been introduced for handling restricted classes of pseudoknots [7-13]. Condon *et al.* [14] give an overview of their structure classes and the algorithm-specific restrictions and Möhl *et al.* [15] develop a general framework showing that all these algorithms follow a general scheme, which they use for efficient alignment of pseudoknotted RNA.

The most general algorithm (with respect to the pseudoknot classes handled) among the above by Rivas and Eddy (R&E) has a running time of $O(n^6)$ time and space consumption of $O(n^4)$. It is therefore too expensive to directly apply this algorithm for large scale data analysis. Unfortunately, even the most efficient algorithm by Reeder and Giegerich (R&G) still has a high running time of $O(n^4)$, although it strongly restricts the class of predictable pseudoknots.

In this paper we introduce the technique of sparsification to the problem of pseudoknotted RNA structure prediction. Sparsification improves the expected running time and space usage of a dynamic programming based structure prediction algorithm without introducing additional restrictions on the structure class handled or compromising the optimality of solutions. Sparsification has been recently applied to improve time and space complexity of various existing RNA-related structure prediction algorithms. In particular, it turned out to be successful for RNA folding for

* Correspondence: backofen@informatik.uni-freiburg.de; cenk@cs.sfu.ca
† Contributed equally
[1]Bioinformatics, Institute of Computer Science, Albert-Ludwigs-Universität, Freiburg, Germany
[2]Lab for Computational Biology, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
Full list of author information is available at the end of the article

pseudoknot-free structures [16,17], simultaneous alignment and folding [18] as well as RNA-RNA interaction prediction [19].

Contributions

We study sparsification of pseudoknotted RNA structure prediction. Algorithms developed for this problem differ from the previously sparsified algorithms by their use of gapped fragments and their more complex recursion structure. Our main contribution in this paper is the solution to the algorithmic challenges due to this increased complexity. Among all DP based pseudoknot prediction algorithms, we focus on the fastest algorithm (R&G) and the most general one (R&E) and develop sparse variants of these dynamic programming algorithms. Furthermore, we consider sparsification of the algorithm by Akutsu *et al.* and Uemura *et al.* (A&U) [9,10] as well as the algorithm by Dirks and Pierce (D&P) [12]. Due to sparsification, the resulting algorithms need to consider only a limited number of candidates substructures compared to the original algorithms. As a result, we analyze the theoretical worst case complexities in terms of the number of candidate substructures. We also present experimental results, comparing our implementations of the original and sparsified *R&G* algorithm. These results suggest a significant (roughly a linear factor) reduction in the number of candidates over the original algorithm.

Methods

Sparsification of the Reeder and Giegerich algorithm

The R&G algorithm [13] predicts the minimum free energy structure allowing canonical pseudoknots for a sequence S of length n. It extends the Zuker algorithm by adding one more matrix K (for knot), where $K(i, j)$ denotes the energy for the best *canonical* pseudoknot that starts at position i and ends at position j. Note that the original presentation of the algorithm in terms of the ADP framework does not explicitly consider a matrix K but only a motif *knot*. Canonical pseudoknots are defined as follows. Each pair of base pairs $p_1 = (i, i')$ and $p_2 = (j', j)$ with $i < j' < i' < j$ induces one canonical pseudoknot that consists of two crossing stems $\{(i, i'), (i+1, i'-1),\ldots, (i+d_{i,i'}-1, i'-d_{i,i'}+1)\}$ and $\{(j', j), (j'+1, j-1),\ldots, (j'+d_{j',j}-1, j-d_{j',j}+1)\}$ where the stacking length of the two stems, $d_{i,i'}$ and $d_{j',j}$, respectively, is maximally extended as long as all base pairs are valid Watson-Crick base pairs.

To allow for sparsification, we restrict the scoring scheme slightly such that the energy of a canonical pseudoknot only depends on the left ends of its base pairs and hence can be described as *PK-Energy*$(i, d_{i,i'}, j', d_{j',j})$. This implies that the scoring scheme does not distinguish between G-C and G-U base pairs in pseudoknot-stems, since their left ends are identical. Then,

$$K(i, j) = \min_{i',j'} score(i, j', i', j) \tag{1}$$

with

$$\begin{aligned} score(i, j', i', j) = \\ &PK\text{›}Energy(i, d_{i,i'}, j', d_{j',j}) + W(i + d_{i,i'}, j' - 1) + \\ &W(j' + d_{j',j}, i' - d_{i,i'}) + W(i' + 1, j - d_{j',j}). \end{aligned} \tag{2}$$

As shown in Figure 1(a), for each canonical pseudoknot starting at i and ending at j the recursion decomposes into the pseudoknot itself and the three fragments in-between its two crossing stems. Such pseudoknots add one case in the computation of a matrix entry $W(i, j)$, which, as in the Zuker algorithm, contains the optimal energy of a substructure starting at position i and ending at position j. Due to the restriction to canonical pseudoknots, the recursion of R&G minimizes only over all possible instances of i' and j', because the maximal stacking lengths $d_{i,i'}$ and $d_{j',j}$ are uniquely determined once i' and j' are fixed. Furthermore, Reeder and Giegerich note that the maximal stacking length $d_{x,y}$ can be precomputed for all x, y in $O(n^3)$ time and stored in an $O(n^2)$ table.

In order to sparsify the algorithm, we develop an appropriate notion of a *candidate* such that it is not necessary to minimize over all possible i' and j' but only over the candidates.

Definition 1 (R&G candidate)

Let $i < j' < i'_1 < i'_2$ *and* $d_{j',j} \le i'_{1'} - j'$ *. Then* i'_1 *dominates* i'_2 *with respect to* $(i, j'\, d_{j',j})$*, iff*

$$\text{score}_{i'_2}(i, j', i'_2) \ge \text{score}_{i'_2}(i, j', i'_1),$$

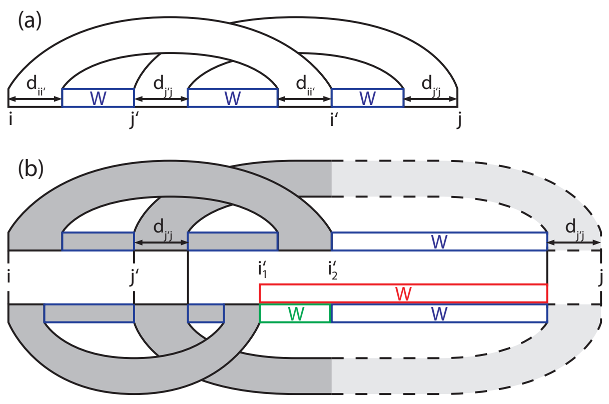

Figure 1 Recursion for canonical pseudoknots (a) and their sparsification (b).

where

$$\begin{aligned}\mathrm{score}_{i_c}(i,j',i') :=&\\ &PK\rangle Energy(i,d_{i,i'},j',d_{j',j}) + W(i+d_{i,i'},j'-1)\\ &+W(j'+d_{j',j},i'-d_{i,i'}) + W(i'+1,i'_c).\end{aligned}$$

We say that i'_2 *is a candidate with respect to* $(i, j', d_{j',j})$ *if there does not exist any* i'_1 *that dominates it.*

The notion of a candidate is visualized in Figure 1(b). There, i'_1 dominates i'_2 if the score for the gray area at the top (including the dashed part whose exact position is not determined) is not better than the score for the corresponding gray area at the bottom plus the green part. Note that these scores (and hence the candidate i') depend only on i, j', and $d_{j',j}$ and are independent of $d_{i,i'}$ and j. The following lemma shows that the notion of a candidate given in Def. 1 is suitable for sparsification, i.e. some i' needs to be considered in the recursion (for all j) only if it is a candidate, because otherwise it is dominated by a candidate that yields a better score.

Lemma 1 (R&G sparsification)

Let i'_2 *be dominated by* i'_1 *with respect to some* $(i, j', d_{j',j})$. *Then for all j it holds* $score(i,j',i'_1,j) \le score(i,j',i'_2,j)$.

Proof We start with the inequality of Def. 1 and add $W(i'_2+1, j-d_{j',j})$ on both sides. Then the claim follows immediately from $W(i'_1+1,j-d_{j',j}) \le W(i'_1+1,i'_2)+W(i'_2+1,j-d_{j',j})$. In Figure 1(b) this corresponds to the fact that the score for the red box is at least as good as the score from the green and the blue box together. This triangle inequality holds by the correctness of the (unsparsified) algorithm: For all $x < y < z$ we have $W(x, y)+W(y+1, z) \le W(x, z)$ since the concatenation of the best structures for the ranges (x, y) and (y, z) always forms a valid structure for the range (x, z) with score $W(x, y)+W(y+1, z)$ which is hence never better than the optimal score $W(x, z)$ for that range. □

The sparsified algorithm maintains lists L_i of candidates for each pair $(j', d_{j',j})$ since only the lists for one i need to be maintained in memory at the same time. Whenever in the computation of some $score(i, j', i', j)$ the i' is considered the first time for this i and j', it is checked whether it is a candidate and if so, it is added to the respective list. For all other instances of j, i' is then considered only if it is contained in the list. The sparsified algorithm is given by the following pseudo-code ($n := |S|$).

```
1: for $i := n$ to 1 do
2:   for all $d_{j',j}, j' \le n$ do
3:     $L_i(j', d_{j',j})$ := empty list;
4:   end for
5:   for $j := i + 3$ to $n$ do
6:     $K(i, j) := \infty$
7:     for $j' := i + 1$ to $j - 2$ do
8:       // check new elements for candidacy
9:       for $i_c := \max\{j'+d_{j',j}, \mathrm{checked}_{i,j',d_{j',j}}+1\}$ to $j - d_{j'j}$ do
10:        if $\mathrm{score}_{i_c}(i,j',i_c) < \mathrm{score}_{i_c}(i,j',i')$ for all $i' \in L_i(j', d_{j',j})$ then
11:          add $i_c$ to $L_i(j', d_{j',j})$
12:        end if
13:      end for
14:      $\mathrm{checked}_{i,j',d_{j',j}} := \max(\mathrm{checked}_{i,j',d_{j',j}}, j-d_{j',j})$
15:      // iterate over all candidates
16:      $K_{i,j',j} := \infty$
17:      for all $i' \in L_i(j', d_{j',j})$ do
18:        $K_{i,j',j} := \min\{K_{i,j',j}, score(i, j', i', j)\}$
19:      end for
20:      $K(i, j) := \min\{K(i, j), K_{i,j',j}\}$
21:    end for
22:    compute matrix entries $V(i, j)$ and $W(i, j)$ as in Wexler et al.
23:      $W(i, j) := \min(W(i, j), K(i, j))$
24:  end for
25: end for
```

The candidate lists are initialized in line 2. In lines 7 to 11 all new values i_c that have not been considered so far, are tested for candidacy. Here, $\mathrm{checked}_{i,j',d_{j',j}}$ denotes the largest i' that has been checked for candidacy in list $L_i(j', d_{j',j})$.

Lines 14 to 17 compute scores $score(i, j', i', j)$ for all candidates i'. In line 20, we compute $W(i, j)$ and $V(i, j)$ as in the sparsified pseudoknot-free structure prediction approach due to Wexler *et al.* [16]. The computation of matrices K and W is interleaved such that all entries $K(i, j)$ and $W(i, j)$ are computed before all entries $K(i', j')$ and $W(i', j')$ for $i \le i' \le j' \le j$ and $i \ne i'$ or $j \ne j'$.

Complexity Analysis

Whereas the original algorithm requires $O(n^4)$ time (for $n = |S|$), the sparsified variant requires $O(n^3L)$ time where L is the total size for all candidate lists of some i i.e. $L := \max_i \sum_{j',d_{j',j}} |L_i(j', d_{j'j})|$. Obviously, $L \le n$. In order to maintain the asymptotic space complexity $O(n^2)$ of the original algorithm, we do not maintain all lists $L_i(j', d_{j',j})$ in memory but only the lists with $d_{j',j} \le k$ where $k > 0$ is a small constant. Please note that to keep presentation simple, we didn't make this explicit in the pseudo-code. Since the maximal stacking length is usually small, there are only very few instances of j with $d_{j',j} > k$ such that for those few j it is cheap to consider all i' as candidates. Hence, we store $O(kn) = O(n)$ candidate lists each requiring at most $O(n)$ space.

Wexler *et al.* [16] use the assumption that RNA folding satisfies the polymer-zeta property to derive a tighter bound on the expected-case asymptotic complexity. However, we focus on the practical speed-up that is obtained by our implementation due to the following reasons. First, it is unclear whether the energy-models for pseudoknot prediction exhibit this property and second it is unclear whether the asymptotic behaviour already appears in the feasible range of input sizes. As shown in the results, the sparsified variant runs two to four times faster than the unsparsified variant for input sizes up to 1000 nucleotides.

Sparsification of the Rivas and Eddy Algorithm

The class of structures predicted by the R&E algorithm [8], here called class of R&E structures, is the most general RNA secondary structure prediction algorithm described in the literature [14]. To keep presentation simple we explain the sparsification strategy for a base-pair maximization algorithm that handles the R&E structure class. Finally, we motivate that sparsification can be transferred to the R&E energy minimization algorithm.

First, we give recursions of base pair maximization for R&E structures. Note that the recursions are intentionally very close to the recursions of the R&E energy minimization algorithm. After initialization for $i \geq j$ and $k \geq l$

$$W(i,j) = \begin{cases} 0 & \text{if } i = j \text{ or } i = j+1 \\ -\infty & \text{if } i > j+1 \end{cases}$$

and

$$\begin{aligned} &W(i,j;k,l) = -\infty \text{ if } j < i \text{ or } l < k \\ &W(i,i;k,k) = \mathrm{bp}(i,k) \end{aligned}$$

Where $\mathrm{bp}(i,j) = \begin{cases} 1 & \text{if } S_i, S_k \text{ complementary} \\ -\infty & \text{otherwise,} \end{cases}$ is the *base pair contribution*, the recursions (R&E recursions) are given for $1 \leq i < j < k < l \leq |S|$ as

$$W(i,j) = \max \begin{cases} W(i,j-1) & (12') \\ \mathrm{bp}(i,j) + W(i+1,j-1) & (1'21') \\ \max_{j'} W(i,j'-1) + W(j',j) & (12) \\ \max_{j',k',l'} \begin{pmatrix} W(i,j'-1;k'+1,l'-1) \\ +W(j',k';l',j) \end{pmatrix} & (1212) \end{cases}$$

$$W(i,j;k,l) = \max \begin{cases} W(i+1,j;k,l) & (1'2G2) \\ W(i,j-1;k,l) & (12'G1) \\ W(i,j;k+1,l) & (1G2'1) \\ W(i,j;k,l-1) & (1G12') \\ \max_{j'} W(i,j') + W(j'+1,j;k,l) & (12G2) \\ \max_{j'} W(i,j'-1;k,l) + W(j',j) & (12G1) \\ \max_{l'} W(i,j;l'+1,l) + W(k,l') & (1G21) \\ \max_{l'} W(i,j;k,l'-1) + W(l',l) & (1G12) \\ \max_{j',k'} \begin{pmatrix} W(i,j'-1;k'+1,l) \\ +W(j',j;k,k') \end{pmatrix} & (12G21) \\ \max_{j',k'} \begin{pmatrix} W(i,j'-1;k,k'-1) \\ +W(j',j;k',l) \end{pmatrix} & (12G12) \\ \max_{k',l'} \begin{pmatrix} W(i,j;k'+1,l'-1) \\ +W(k,k';l',l) \end{pmatrix} & (1G212) \\ \max_{i',j'} \begin{pmatrix} W(i,i'-1;j'+1,j) \\ +W(i',j';k,l) \end{pmatrix} & (121G2). \end{cases}$$

It is easy to check that $W(1, |S|)$ is the maximal number of base pairs in a *R&E* structure of S, because the recursions perform the same decompositions as the original R&E recursions. Note that $W(i, j; k, l)$ is the maximal number of base pairs in structures with at least one base pair that spans the gap. We label each recursion case in a way that illustrates the type of the decomposition of this case. The idea of these labels is taken from Möhl *et al.* [15], where we developed a type system for decompositions, which there are called splits. For this reason, we call these labels split types, however, we won't need any details of the typing system. The decomposition by R&E is illustrated in Figure 2.

A *fragment* is defined as a set of positions of the fixed sequence S. The fragments corresponding to matrix

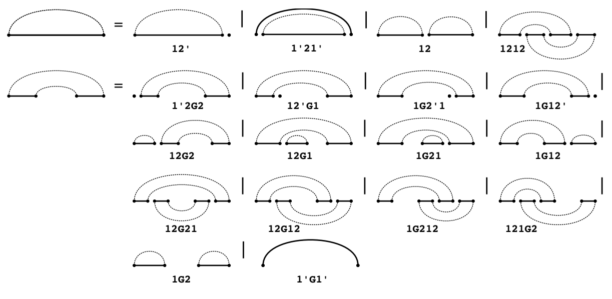

Figure 2 Decomposition for R&E base pair maximization annotated with labels, i.e. split types, of the corresponding recursion cases.

entries in the *R&E* recursion can be described conveniently by their boundaries. We distinguish *ungapped fragments* $F = \{i,...,j\}$, written (i, j), and *1-gap fragments* $F' = \{i,...,j\} \cup \{k,...,l\}$, written $(i, j; k, l)$ where i, j, k, l, are called *boundaries* of respective F or F'. A *split* of a fragment F is a tuple (F_1, F_2) such that $F = F_1 \cup F_2$ and $F_1 \cap F_2 \varnothing$.

For our sparsification approach, we will show that in each recursion case, certain optimally decomposable fragments do not have to be considered for computing an optimal solution, because each decomposition using these fragments can be replaced by a decomposition using a smaller fragment. We define optimal decomposability with respect to the split type of a R&E recursion case.

Definition 2 (Optimally decomposable)

A fragment F is optimally decomposable by a split of type T (T-OD) *iff there is a split (F_1, F_2) that occurs in recursion case T and $W(F_1) + W(F_2) \geq W(F)$.*

A fragment F is optimally decomposable w.r.t a set of split types $\mathcal{T}$ ($\mathcal{T}$-OD) *iff F is T-OD for some $T \in \mathcal{T}$.*

Here, we emphasize that testing T-OD for a fragment F is simple in a run of the DP algorithm. After evaluating the case T in the computation of $W(F)$, one compares the maximum of the case to $W(F)$. For example, a fragment $(i, j; k, l)$ is 12G21-OD iff $W(i, j; k, l) = \max_{j', k'} W(i, j' - 1; k' + 1, l) + W(j', j; k, k')$.

In the following we show that for the maximization in a recursion case T, we do not need to consider T'-OD fragments as second fragment of the split, where T' is from a T-specific set of split types. As an example consider the recursion case 12G21, which splits fragments $(i, j; k, l)$ into $F_1 = (i, j' - 1; k' +1, l)$ and $F_2 = (j', j; k, k')$. Assume that F_2 is 12G21-OD. Then we can show that every evaluation of $W(F)$ where $W(F) = W(F_1) + W(F_2)$ can be replaced by another at least equally good evaluation that splits F into F_1' and $F_2' \subset F_2$, where F_2' is the second fragment in the 12G21-split of F_2. However, note that the argument is split type specific and cannot be applied e.g. when F_2 is 12G12-OD.

For sparsifying R&E, we define the following sets of split types.

$$\begin{aligned}
\mathcal{T}_{12}^{\mathrm{RE}} &= \{12\} \\
\mathcal{T}_{1212}^{\mathrm{RE}} &= \{12\mathrm{G}2,\ 12\mathrm{G}1,\ 1\mathrm{G}21\} \\
\mathcal{T}_{12\mathrm{G}1}^{\mathrm{RE}} = \mathcal{T}_{1\mathrm{G}12}^{\mathrm{RE}} &= \mathcal{T}_{1\mathrm{G}21}^{\mathrm{RE}} = \{12\} \\
\mathcal{T}_{12\mathrm{G}2}^{\mathrm{RE}} &= \{12\mathrm{G}2\} \\
\mathcal{T}_{12\mathrm{G}21}^{\mathrm{RE}} &= \{12\mathrm{G}2,\ 1\mathrm{G}12,\ 12\mathrm{G}21\} \\
\mathcal{T}_{12\mathrm{G}12}^{\mathrm{RE}} &= \{12\mathrm{G}2,\ 1\mathrm{G}21,\ 12\mathrm{G}12\} \\
\mathcal{T}_{1\mathrm{G}212}^{\mathrm{RE}} &= \{12\mathrm{G}1,\ 1\mathrm{G}21,\ 12\mathrm{G}21\} \\
\mathcal{T}_{121\mathrm{G}2}^{\mathrm{RE}} &= \{12\mathrm{G}2,\ 12\mathrm{G}1,\ 121\mathrm{G}2\}
\end{aligned}$$

These sets are defined such that in a recursion case T, whenever the second fragment of a split (F_1, F_2) of F can be optimally decomposed by a split of a type in $\mathcal{T}_T^{\mathrm{RE}}$, a different split (F_1', F_2') of type T can be applied to F, where $F_2' \subset F_2$. As we show later, this split will be just as good as (F_1, F_2) for computing $W(F)$.

Then, one systematically obtains sparsified recursion equations $W'(i, j)$ and $W'(i, j; k, l)$ from the equations for $W(i, j)$ and $W(i, j; k, l)$ by replacing symbol W by W' and modifying them in the following way. For each case T in the recursion of $W(i, j)$ and $W(i, j; k, l)$ that maximizes over $W(F_1)+W(F_2)$ for respective splits of the fragment $F = (i, j)$ or $F = (i, j; k, l)$, maximize only over fragments F_2 that are not $\mathcal{T}_T^{\mathrm{RE}}$-OD. In an algorithm that evaluates the sparsified recursion, such non-$\mathcal{T}_T^{\mathrm{RE}}$-OD fragments correspond to entries of candidate lists. For example, case 12G21 of W is modified in the equation for $W'(i, j, k, l)$ to

$$\max_{\substack{j',k',(j',j;k,k') \\ \text{not } \mathcal{T}_{12\mathrm{G}21}^{\mathrm{RE}}\text{-OD}}} \begin{pmatrix} W'(i, j'-1; k'+1, l) \\ +W'(j', j; k, k') \end{pmatrix} \quad (12\mathrm{G}21 \text{ of } \mathbf{W}').$$

Theorem 1

Let W be the matrix of the R&E recursion and W' its sparsified variant, then $W(1, |S|) = W'(1, |S|)$.

Proof We show for all $1 \leq i, j, k, l \leq |S|$, $W(i, j) = W'(i, j)$ and $W(i, j, k, l) = W'(i, j; k, l)$. First note that it holds that $W(i, j) \geq W'(i, j)$ and $W(i, j; k, l) \geq W'(i, j; k, l)$. The claim is shown by induction on the fragment size and a case distinction over recursion cases. For the case of split type 12, we show that

$$\begin{aligned}
&\max_{j'} W(i, j'-1) + W(j', j) = \\
&\max_{j',\,(j',j) \text{ not } \mathcal{T}_{12}^{\mathrm{RE}}\text{-OD}} W'(i, j'-1) + W'(j', j).
\end{aligned}$$

Let (j', j) be 12-OD for some $j' : i \leq j' \leq j$. By IH, it suffices to find a (smaller) fragment (j'', j), where $j'' > j$ and $W(i, j'' - 1) + W(j'', j) \geq W(i, j' - 1) + W(j', j)$. Either (j', j) is not 12-OD or there is a j'', such that $W(j', j) = W(j', j'' - 1) + W(j'', j)$ and thus $W(i, j'' - 1)+W(j'', j) \geq W(i, j' - 1)+W(j', j)$ because

$$\begin{aligned}
&W(i, j''-1) + W(j'', j) \\
\geq_{\Delta\text{-ineq}}\ &W(i, j'-1) + W(j', j''-1) + W(j'', j) \\
=_{12\text{-OD}}\ &W(i, j'-1) + W(j', j).
\end{aligned}$$

The triangle inequality (Δ-ineq) is an immediate consequence of the correctness of the recursion for W. Thus, for the decompositions of all recursion cases there holds such a corresponding inequation. Analogous arguments can be given for all other modified recursion cases. Exemplarily, we elaborate the argument for the complex case 12G21. Let $F_1 = (i, j' - 1; k' + 1, l)$ and $F_2 = (j', j; k, k')$, such that (F_1, F_2) is a split of type 12G21 of $(j, j; k, k)$. We need to show for all $\mathcal{T}^{\mathrm{RE}}_{12\mathrm{G}21}$-OD fragments F_2 there are non-empty ungapped or 1-gap fragments F'_1 and F'_2, where $F'_1 \cup F'_2 = F_2, F'_1 \cap F'_2 = \emptyset$, and $W(F_1 \cup F'_1) + W(F'_2) \geq W(F_1) + W(F_2)$ and the split $(F_1 \cup F'_1, F'_2)$ occurs in a recursion case of R&E. Again, either F_2 is not $\mathcal{T}^{\mathrm{RE}}_{12\mathrm{G}21}$-OD or one of the following cases applies. Case 1 (12G2): for some j'', $W(j', j; k, k') = W(j', j'' - 1) + W(j'', j; k, k')$. Then, the claim holds for $F'_1 = (j', j'' - 1)$ and $F'_2 = (j'', j; k, k')$ by triangle inequality and split $(F_1 \cup F'_1, F'_2)$ occurs in recursion case 12G21. Case 2 (2G21): for some k'', $W(j', j; k, k') = W(j', j; k, k'') + W(k'' + 1, k')$. The claim holds for $F'_2 = (j', j; k, k'')$. Case 3 (12G21): for some j'', k'', $W(j', j; k, k') = W(j', j'' - 1; k'' + 1, k') + W(j'', j; k, k'')$. Again, this satisfies the claim by triangle inequality.

Algorithm

The recursion equation W' tailors a sparsified dynamic programming algorithm for the evaluation of $W'(1, |S|)$ with very limited overhead. We maintain separate candidate lists for each sparsified recursion case. As already mentioned, the T-OD properties of each fragment F can be easily checked after evaluation of each case of $W(F)$. A fragment is added to a candidate list for recursion case T iff it is not $\mathcal{T}^{\mathrm{RE}}_{T}$-OD. The maximizations are restricted to run only over the candidates in the respective candidate list. Their intended use dictates the exact nature of such candidate lists. For a case T, which splits a fragments T into T_1 and T_2, there are candidate lists for all boundaries of a fragment T_2 that are not adjacent to boundaries of T_1 due to split type T. The list entries are tuples of the adjacent boundaries and the fragment score for T_2. In order to profit from a reduced number of candidates in space, we maintain two three-dimensional slices of the matrix for $W(i, j; k, l)$, storing entries only for the current i and $i + 1$. Scores $W(i, j; k, l)$ for larger i are stored for candidates only. Pseudocode of the sparsified algorithm is given in Figure 3.

R&E Free Energy Minimization

Sparsification is analogously applied to the energy minimizing R&E algorithm. This algorithm distinguishes several additional matrices that contain minimal energies for fragments (i, j) or $(i, j; k, l)$ under the condition that respectively the base pair (i, j) or base pairs (i, l) and (j, k) or one of them exist. Almost all decompositions in the recursion for these matrices are of discussed split types and are sparsified analogously. The only notable exception is due to internal loops. Internal loops require minimizing over all possible positions of the inner loop base pair, where commonly the loop size is restricted by a constant K such that minimizing takes constant time. However, handling inner loops requires access to entries of non-candidate fragments $(i', j'; k', l')$ for $i \leq i' \leq i + K + 2$. This is handled by maintaining matrix slices for i to $i + K + 2$ in $O(n^3)$ space, which preserves total space complexity.

Complexity Analysis

The described algorithm profits from sparsification in time and space. Compared to $O(n^6)$ time and $O(n^4)$ space of the unsparsified algorithm (for $n = |S|$), we obtain complexities in the number of candidates. Let Z_T denote the maximal length of a candidate lists for case T and Z denote the total number of entries in all lists. Then, the time complexity is $O(n^2(Z_{12} + Z_{1212}) + n^4(Z_{12G2} + Z_{12G1} + Z_{1G21} + Z_{1G12} + Z_{12G21} + Z_{12G12} + Z_{1G212} + Z_{121G2}))$ and space complexity is $O(n^3 + Z)$. In the worst case, Z_{12}, Z_{12G2}, Z_{12G1}, Z_{1G21} and Z_{1G12} are $O(n)$, Z_{12G21}, Z_{12G12}, Z_{1G212}, Z_{121G2} are $O(n^2)$, and Z_{1212} is $O(n^3)$, finally Z is $O(n^4)$ in the worst case.

Sparsification of the Dirks and Pierce Algorithm

Dirks and Pierce [12] present a pseudoknot prediction algorithm that takes $O(n^5)$ time and $O(n^4)$ space. Note that whereas Dirks and Pierce present their decomposition for computing the partition function, we sparsify the corresponding minimum free energy prediction algorithm. As mentioned in [15] this algorithm can be considered as a restriction of the algorithm by Rivas and Eddy to the cases

$$\begin{array}{cccccc} 12' & 1'2G2 & 12'G1 & 1G2'1 & 1G12' & \text{and} \\ 12 & 1212 & 12G2 & 12G1 & 1G21 & 1G12 \end{array}$$

with an additional case 1'2G21' that composes a gapped fragment $(i, j; k, l)$ from a single base pair (i, l) and $(i + 1, j; k, l - 1)$.

The non-constant cases 12, 1212, 12G2, 12G1, 1G21, and 1G12 can be sparsified exactly as the corresponding cases of the Rivas and Eddy algorithm with the following sets of split types:

$$\mathcal{T}^{\mathrm{DP}}_{12} = \{12\} \qquad \mathcal{T}^{\mathrm{DP}}_{1212} = \{12\mathrm{G}2, 12\mathrm{G}1, 1\mathrm{G}21\}$$
$$\mathcal{T}^{\mathrm{DP}}_{12\mathrm{G}2} = \{12\mathrm{G}2\} \quad \mathcal{T}^{\mathrm{DP}}_{12\mathrm{G}1} = \mathcal{T}^{\mathrm{DP}}_{1\mathrm{G}21} = \mathcal{T}^{\mathrm{DP}}_{1\mathrm{G}12} = \{12\}$$

Note that the additional case 1'2G21' does not need to be sparsified, because it is computed in constant time. Analogously to our discussion of the R&E algorithm, one obtains space and time complexities of the sparsified algorithm in terms of the length of candidate lists and the total number of candidates.

```
initialize all candidate lists $L$ as empty
for i:=n to 1 do
  W[i][i-1]:=0
  for j:=i to n do
    $W_{12'} := W[i][j-1]$;    $W_{1'21'} := W[i+1][j-1] + \mathrm{bp}(i,j)$
    $W_{12} := \max_{(j',w)\in L(j,12)} W[i][j'-1] + w$
    $W_{1212} := \max_{(j',k',l',w)\in L(j,1212)} W[j'][k'][l'] + w$
    $W := \max\{W_{12'}, W_{1'21'}, W_{12}, W_{1212}\}$
    if $W_{12} < W$ then
      push L(j,12), (i,W);    push L(j,1G21), (i,W)
      push L(i,12G1), (j,W);    push L(i,1G12), (j,W)
    end if
    W[i][j]:=W
    initialize W[j][k][l]
    for k:=n to j+2 do
      for l:=k to n do
        $W_{1'2G2} := W1[j][k][l]$;    $W_{1'2G1} := W[j-1][k][l]$
        $W_{1G2'1} := W[j][k+1][l]$;    $W_{1G12'} := W[j][k][l-1]$
        $W_{12G2} := \max_{(j',w)\in L(j,k,l,12G2)} W[j'-1] + w$
        $W_{12G1} := \max_{(j',w)\in L(j,12G1)} W[j'-1][k][l] + w$
        $W_{1G21} := \max_{(k',w)\in L(j,1G21)} W[j][k'+1][l] + w$
        $W_{1G12} := \max_{(l',w)\in L(j,1G12)} W[j][k][l'-1] + w$
        $W_{12G21} := \max_{(j',k',w)\in L(j,k,12G21)} W[j'-1][k'+1][l] + w$
        $W_{12G12} := \max_{(j',k',w)\in L(j,l,12G12)} W[j'-1][k][k'-1] + w$
        $W_{1G212} := \max_{(k',l',w)\in L(k,l,1G212)} W[j][k'+1][l'-1] + w$
        $W_{121G2} := \max_{(i',j',w)\in L(k,l,1G212)} W[i'-1][j'+1][j] + w$
        $W := \max\{W_{1'2G2}, W_{1'2G1}, W_{1G2'1}, W_{1G12'}, W_{12G2}, W_{12G1}, W_{1G21}, W_{1G12},$
                  $W_{12G21}, W_{12G12}, W_{1G212}, W_{121G2}\}$
        if $\forall T \in \mathcal{T}^{\mathrm{RE}}_{1212} : W_T < W$ then push $L(j,1212)$, $(i,j,k,W)$
        if $\forall T \in \mathcal{T}^{\mathrm{RE}}_{12G2} : W_T < W$ then push $L(j,k,l,12G2)$, $(i,W)$
        if $\forall T \in \mathcal{T}^{\mathrm{RE}}_{12G21} : W_T < W$ then push $L(j,k,12G21)$, $(i,l,W)$
        if $\forall T \in \mathcal{T}^{\mathrm{RE}}_{12G12} : W_T < W$ then push $L(j,l,12G12)$, $(i,k,W)$
        if $\forall T \in \mathcal{T}^{\mathrm{RE}}_{1G212} : W_T < W$ then push $L(i,l,1G212)$, $(j,k,W)$
        if $\forall T \in \mathcal{T}^{\mathrm{RE}}_{121G2} : W_T < W$ then push $L(k,l,121G2)$, $(i,j,W)$
        W[j][k][l] := W
      end for
    end for
  end for
  for all $1 \le j < k \le l \le n$ do $W1[j][k][l] := W[j][k][l]$
end for
```

Figure 3 Pseudocode for R&E-style base pair maximization.

Sparsification of the Akutsu and Uemura Algorithm

In this section we consider the pseudoknot prediction algorithm that was developed by Uemura *et al.* [9] based on tree adjoining grammars and later reformulated by Akutsu *et al.* [10] as dynamic programming algorithm. The algorithm predicts simple pseudo-knots in $O(n^4)$ time and $O(n^3)$ space. It can also be considered as a restriction of the algorithm by Rivas and Eddy. It is restricted to splits of the following types (again following the typing scheme of [15]):

$$\begin{array}{cccc} 12 & 121 & 12'G2'1 & 1G2'12' \\ 12'G1 & 1G2'1 & 1G12' & 12'G2' \end{array}$$

and ommitted trivial, constant cases. Compared to the R&E algorithm, all cases that dominate the complexity are restricted to have only one possible split per instance (as indicated by the ‘ symbols; confer the additional case/split type of the algorithm by Dirks and Pierce). All non-constant cases, i.e. the first two rules, can still be sparsified analogous to sparsification of the algorithm of Rivas and Eddy using split type sets

$$\mathcal{T}_{12}^{\mathrm{AU}} = \{12\} \quad \text{and} \quad \mathcal{T}_{121}^{\mathrm{AU}} = \{12,121\}.$$

The restriction introduced by Akutsu and Uemura could be considered as a very simple, static form of sparsification. For each fragment annotated with symbol ', only one candidate (namely the smallest possible one) is considered. In contrast to sparsification as it is discussed in this paper, Akutsu's and Uemura's modification of the R&E algorithm reduces the worst-case complexity at the price of restricting the class of pseudoknots.

Results and Discussion

In order to evaluate the effect of sparsification on pseudoknotted RNA secondary structure prediction, we implemented original and sparsified variants of the Reeder and Giegerich (R&G) algorithm.

Data Set

We obtained all RNA sequences from Pseu-doBase [20], which are known to have some pseudo-knots in their secondary structures. This set contains 294 sequences that their length is distributed between 76 nt and 93399 nt. We randomly divided all long sequences into subsequences shorter than 1000 nt. Therefore the data set that we used in our experiments contains 1563 sequences with length between 76 nt and 1000 nt.

Performance

We applied both variants of the R&G algorithm to our data set. Figure 4 shows the running time of the algorithms on a server with Intel Core Duo CPU at 2.53 GHz and 4 GB RAM. The results in Figure 4 show that sparsification significantly improves the running time of the R&G algorithm. As the RNA sequences get longer, the relative performance of the sparsified algorithm (with respect to the non-sparsified ones) improves. Figure 4(b) shows the speedup of the sparsified algorithm, which fits well to a linear regression (R^2 = 0.84).

Number of candidates

For a better understanding of the effect of sparsification on the R&G algorithm, we measured the number of (i', j') pairs which are checked in each fragment $[i, j]$ in both original and sparsified variants of the algorithm. Note that the number of (i', j') pairs is in order of $O((j - i)^2)$ in the worst case. Figure 5 shows the average number of (i', j') pairs on fragments of equal length which are checked by the two variants of the algorithm. As expected, this amount is significantly smaller for the sparsified algorithm compared to the original one. Moreover, we observe that as the fragments get longer, the difference between the average number of (i', j') pairs in the sparsified and the original algorithm increases. We define the work load per each fragment $[i, j]$ as the number of candidate (i', j') pairs. Figure 5(b), shows a significant reduction of the work load in the sparsified algorithms. As it can be seen for subsequences of length 1000 nt, the work load by the sparsified algorithm is reduced by a factor of about 10 compared to the original algorithm. Note that the work load reduction at fragment length 1000 nt does not yield the same speedup for sequences of length 1000 nt (here this speedup is about 3.5, confer Figure 4(b)), because for a sequence of length n, all fragments of smaller length are processed by the algorithm.

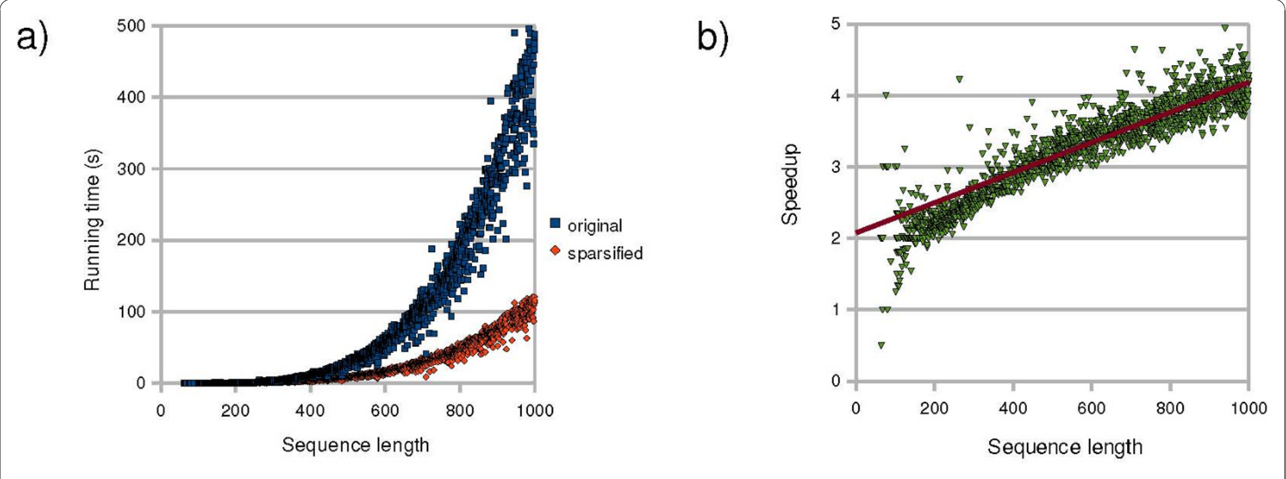

Figure 4 Running times of the original and sparsified variants of the R&G algorithm.

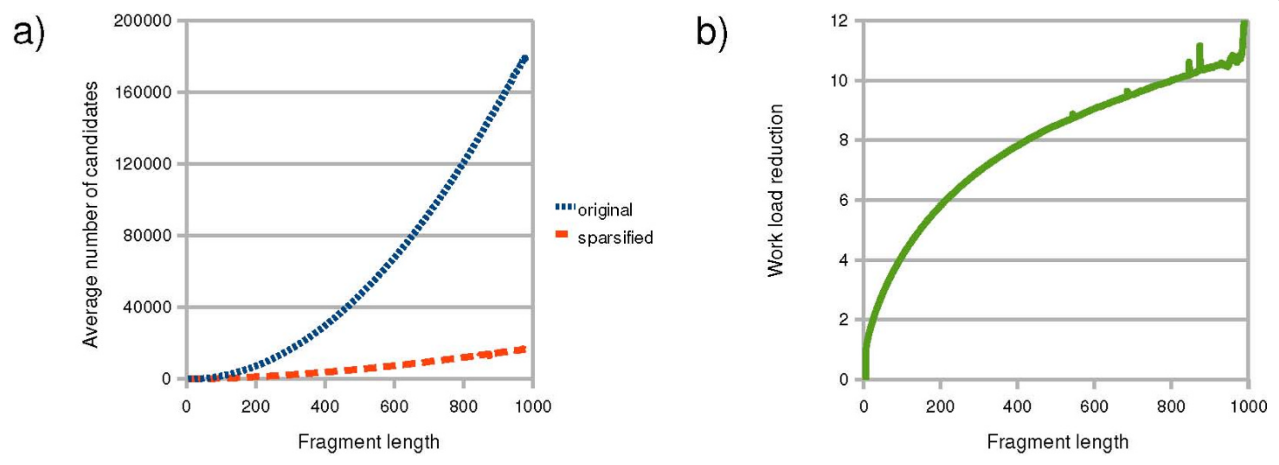

Figure 5 Average number of (*i'*, *j'*) candidates in the original and sparsified variants of the R&G algorithm.

Conclusions

The presented work gives four examples for sparsification in the context of gap fragments and a complex recursion structure. We successfully sparsified the fastest and the most complex pseudo-knot structure prediction algorithm for RNA, as well as two algorithms with intermediate complexity. Since sparsification is similar in all these algorithms, the paper motivates further generalization of sparsification for systematic application to complex DP-algorithms as RNA structure prediction algorithms. Even more, by providing detailed examples the paper directly suggests such generalization. Our results from an implementation of the sparsified Reeder and Giegerich algorithm show a significant, presumably even linear, expected work load reduction due to sparsification. As future work, it would be interesting to develop optimizations for the partition function based variants of pseudoknot prediction where sparsification is not directly applicable.

Acknowledgements

This work is partially supported by DFG grants WI 3628/1-1, EXC 294, and BA 2168/3-1. R. Salari was supported by SFU-CTEF funded Bioinformatics for Combating Infectious Diseases Project co-lead by S.C. Sahinalp. S.C. Sahinalp was supported by MITACS, NSERC, the CRC program and the Michael Smith Foundation for Health Research.

Author details

[1]Bioinformatics, Institute of Computer Science, Albert-Ludwigs-Universität, Freiburg, Germany. [2]Lab for Computational Biology, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. [3]Computation and Biology Lab, CSAIL, MIT, Cambridge MA, USA. [4]Centre for Biological Signalling Studies (bioss), Albert-Ludwigs-Universität, Freiburg, Germany.

Authors' contributions

All authors developed the ideas for this project. MM, RS, and SW elaborated the technical contribution and wrote the paper. RS did the implementation and evaluation. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Received: 27 October 2010 Accepted: 31 December 2010
Published: 31 December 2010

References

1. Sharp PA: **The centrality of RNA.** *Cell* 2009, **136(4)**:577-80.
2. Amaral PP, Dinger ME, Mercer TR, Mattick JS: **The eukaryotic genome as an RNA machine.** *Science* 2008, **319(5871)**:1787-9.
3. Washietl S, Pedersen JS, Korbel JO, Stocsits C, Gruber AR, Hackermuller J, Hertel J, Lindemeyer M, Reiche K, Tanzer A, Ucla C, Wyss C, Antonarakis SE, Denoeud F, Lagarde J, Drenkow J, Kapranov P, Gingeras TR, Guigo R, Snyder M, Gerstein MB, Reymond A, Hofacker IL, Stadler PF: **Structured RNAs in the ENCODE selected regions of the human genome.** *Genome Res* 2007, **17(6)**:852-64.
4. Mattick JS, Makunin IV: **Non-coding RNA.** *Hum Mol Genet* 2006, **15(Spec No 1)**:R17-29.
5. Staple DW, Butcher SE: **Pseudoknots: RNA structures with diverse functions.** *PLoS Biol* 2005, **3(6)**:e213.
6. Xayaphoummine A, Bucher T, Thalmann F, Isambert H: **Prediction and statistics of pseudoknots in RNA structures using exactly clustered stochastic simulations.** *Proc Natl Acad Sci USA* 2003, **100(26)**:15310-5.
7. Lyngso RB, Pedersen CNS: **Pseudoknots in RNA Secondary Structures.** *Proceedings of the Fourth Annual International Conferences on Computational Molecular Biology* ACM Press; 2000.
8. Rivas E, Eddy SR: **A dynamic programming algorithm for RNA structure prediction including pseudoknots.** *J Mol Biol* 1999, **285(5)**:2053-68.
9. Uemura Y, Hasegawa A, Kobayashi S, Yokomori T: **Tree adjoining grammars for RNA structure prediction.** *Theor Comput Sci* 1999, **210**:277-303.
10. Akutsu T: **Dynamic programming algorithms for RNA secondary structure prediction with pseu-doknots.** *Discrete Appl Math* 2000, **104**:45-62.
11. Deogun JS, Donis R, Komina O, Ma F: **RNA secondary structure prediction with simple pseudoknots.** *Proceedings of the second conference on Asia-Pacific bioinformatics* Darlinghurst, Australia, Australia: Aus-tralian Computer Society, Inc.; 2004, 239-246.
12. Dirks RM, Pierce NA: **A partition function algorithm for nucleic acid secondary structure including pseudoknots.** *J Comput Chem* 2003, **24(13)**:1664-77.
13. Reeder J, Giegerich R: **Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics.** *BMC Bioinformatics* 2004, **5**:104.
14. Condon A, Davy B, Rastegari B, Zhao S, Tarrant F: **Classifying RNA pseudoknotted structures.** *Theor Comput Sci* 2004, **320**:35-50.
15. Möhl M, Will S, Backofen R: **Lifting prediction to alignment of RNA pseudoknots.** *J Comput Biol* 2010, **17(3)**:429-42.
16. Wexler Y, Zilberstein CBZ, Ziv-Ukelson M: **A Study of Accessible Motifs and RNA Folding Complexity.** In *Proceedings of the Tenth Annual International Conferences on Computational Molecular Biology, Volume 3909 of Lect Notes*

Comput Sci. Edited by: Apostolico A, Guerra C, Istrail S, Pevzner PA, Waterman MS. Springer; 2006:473-487.

17. Backofen R, Tsur D, Zakov S, Ziv-Ukelson M: **Sparse RNA Folding: Time and Space Eficient Algorithms.** In *Proceedings of the 20th Symposium on Combinatorial Pattern Matching, Volume 5577 of Lect Notes Comput Sci.* Edited by: Kucherov G, Ukkonen E. Springer; 2009:249-262.
18. Ziv-Ukelson M, Gat-Viks I, Wexler Y, Shamir R: **A Faster Algorithm for RNA Co-folding.** In *Proceedings of the 8th Workshop on Algorithms in Bioinformatics, Volume 5251 of Lect Notes Comput Sci.* Edited by: Crandall KA, Lagergren J. Springer; 2008:174-185.
19. Salari R, Möhl M, Will S, Sahinalp S, Backofen R: **Time and Space Efficient RNA-RNA Interaction Prediction via Sparse Folding.** In *Proceedings iof the Fourteenth Annual International Conferences on Computational Molecular Biology, Volume 6044 of Lect Notes Comput Sci.* Edited by: Berger B. Springer Berlin/Heidelberg; 2010:473-490.
20. van Batenburg FH, Gultyaev AP, Pleij CW, Ng J, Oliehoek J: **PseudoBase: a database with RNA pseudoknots.** *Nucleic Acids Res* 2000, **28**:201-4.

doi:10.1186/1748-7188-5-39
Cite this article as: Möhl *et al.*: **Sparsification of RNA structure prediction including pseudoknots.** *Algorithms for Molecular Biology* 2010 **5**:39.