Rampášek *et al. BMC Bioinformatics* (2016) 17:216
DOI 10.1186/s12859-016-1074-x

BMC Bioinformatics

METHODOLOGY ARTICLE Open Access

RNA motif search with data-driven element ordering

Ladislav Rampášek[1,2,3], Randi M. Jimenez[2], Andrej Lupták[2*], Tomáš Vinař[3] and Broňa Brejová[3*]

Abstract

Background: In this paper, we study the problem of RNA motif search in long genomic sequences. This approach uses a combination of sequence and structure constraints to uncover new distant homologs of known functional RNAs. The problem is NP-hard and is traditionally solved by backtracking algorithms.

Results: We have designed a new algorithm for RNA motif search and implemented a new motif search tool RNArobo. The tool enhances the RNAbob descriptor language, allowing insertions in helices, which enables better characterization of ribozymes and aptamers. A typical RNA motif consists of multiple elements and the running time of the algorithm is highly dependent on their ordering. By approaching the element ordering problem in a principled way, we demonstrate more than 100-fold speedup of the search for complex motifs compared to previously published tools.

Conclusions: We have developed a new method for RNA motif search that allows for a significant speedup of the search of complex motifs that include pseudoknots. Such speed improvements are crucial at a time when the rate of DNA sequencing outpaces growth in computing. RNArobo is available at http://compbio.fmph.uniba.sk/rnarobo.

Keywords: RNA motif search, Pseudoknot, Search order, Entropy

Background

Functional RNAs are often more conserved in their structure than in sequence. Thus to find RNAs related to a known example, we look for sequences capable of assuming the appropriate secondary structure. Here, we investigate the problem of RNA motif search based on user-defined descriptors. RNA motif descriptors specify restrictions on base-pairing structure of the target RNA, as well as sequence constraints characterizing conserved functional sites. As opposed to popular fully-automated systems based on probabilistic models [1–3], this approach allows expert users to handcraft motif descriptors and highlight the most important features of the target RNAs, thus better targeting a particular biological phenomenon [4–7]. In this paper, we revisit the problem of descriptor-based search and present a new tool, RNArobo, that improves the speed of such searches compared to previous methods, including RNAbob [5], RNAMotif [8], RNAMot [4] and RaligNAtor [9].

Currently, most popular tools for RNA motif search, such as InferRNAl [3] or CMFinder [2], are not based on handcrafted motif descriptors. Instead, they use covariance models founded on stochastic context-free grammars, that are built automatically from a set of known occurrences of the target RNA. This approach addresses many shortcomings of descriptor-based methods, most notably the difficulties in deciding which parts of the motif are important for recognizing a particular RNA, as well as high false positive rates of less specific descriptors.

Covariance models are relatively rich probabilistic models, and consequently many examples are required to build a model of a given RNA family. This precondition can be sometimes easily satisfied, most notably in cases where an alignment of the target family is already present in a database, such as Rfam [10].

However, if only a few examples are known for a particular RNA motif, we are under the necessity to find more occurrences before such parameter-rich models can be employed. In such cases, motif descriptors have been used

*Correspondence: aluptak@uci.edu; brejova@dcs.fmph.uniba.sk
[2]Department of Pharmaceutical Sciences, Chemistry, and Molecular Biology and Biochemistry, University of California, Irvine, 2141 Natural Sciences 2, Irvine, CA, 92697, USA
[3]Faculty of Mathematics, Physics, and Informatics, Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia
Full list of author information is available at the end of the article

with great success to uncover the distribution of small structured RNAs in the genomic space. These functional RNAs include hammerhead ribozymes [11–17], hepatitis delta virus(HDV)-like ribozymes [7, 18–20], as well as genomic aptamers, including the first known human aptamers [21].

This approach is particularly useful for searching for structures that are hard to predict from simple thermodynamic models, such as pseudoknots and nested multi-pseudoknots. HDV-like ribozymes, which have only five conserved, non-contiguous nucleotides out of approximately 50 necessary to form the minimal catalytically-proficient double-pseudoknot [18, 20], represent a particularly striking example of a functional RNA with low sequence conservation and strict structural requirements. Loose descriptors with low sequence requirements tend to yield large numbers of matches in low-complexity genomic sequences (such as long AT and GT repeats); on the other hand, overly strict descriptors often yield too few or no examples of the sought-for functional RNA. To maximize the yield of bona fide examples of functional RNAs with low sequence requirements, their motif descriptors require careful tuning and multiple runs through available genomic sequences. There is thus a great need for efficient descriptor-based search algorithms.

The specific search problem addressed by our method is NP-hard [22]; hardness was also proved for other similar problems involving alignment with arbitrary non-nested interactions [23, 24]. On the other hand, structures without pseudoknots or with simple pseudoknot configurations can be solved by dynamic programming in polynomial time [6, 24, 25], but the running time is at least cubic in the size of the sequence. Nevertheless, even algorithms with worst-case exponential time were shown to be effective in practice, such as backtracking algorithm of RNAMot [4] or non-deterministic finite-state automata with node rewriting of RNAbob [5]. Many other tools were subsequently developed, including Locomotif [6], Palingol [26], RNAMotif [8], PatSearch [27], and RNAMST [28]. Individual tools differ in descriptor capabilities and post-processing options; an extensive review can be found in [29].

To speed up the search, some tools use advanced data structures to build an index of target DNA. For example, Structator [30] and RNAPattMatch [31] use affix arrays [32] and RaligNAtor [9] uses enhanced suffix arrays [33].

Here, we present a new tool, RNArobo, which builds on the descriptor format of RNAbob and the backtracking algorithm of RNAMot. We improve these tools in two ways. First, we extend the RNAbob descriptor format to allow insertions representing bulges in helical elements. In our experiments, we demonstrate that this seemingly minor change helps to better characterize certain families of ribozymes and aptamers and even enables discovery of new occurrences of these motifs that are likely biologically active. Second, we developed a new method for improving the running time of the backtracking algorithm, in some cases speeding up motif searches more than 100-fold compared to other tools.

Each RNA structure descriptor consists of several structural elements. In our algorithm, individual elements are aligned to the DNA sequence by dynamic programming, with backtracking guiding the search for successive elements to appropriate locations with respect to the already matched elements.

The performance of backtracking depends greatly on ordering of elements in the search. Ideally, the first elements will have few matches, filtering out most of the sequence from further processing. Such filtering is a common theme in many text search methods, such as the popular sequence similarity search tool BLAST [34]. Finding the best element ordering for the backtracking search is an interesting and non-trivial problem, due to complex dependencies between locations of individual elements. We approach this as an on-line problem, using the observed performance of the search so far to adjust the ordering on-the-fly. We demonstrate that this strategy leads to a significant reduction in the running time on real data, especially for complex descriptors.

The rest of the paper is organized as follows. First, we define descriptors and their capabilities, and describe the basic backtracking algorithm. Then we introduce our data-driven element ordering strategy. Finally, we demonstrate effectiveness of our approach by revisiting the results of several biological studies and compare our running time with several existing tools. The software tool RNArobo implementing improvements described in this paper is available at http://compbio.fmph.uniba.sk/rnarobo.

Methods

Descriptor-based search for RNA motifs

Here, we briefly describe the search algorithm implemented in our tool RNArobo which is loosely based on the algorithm of RNAMot [4]. The input for the algorithm is a *descriptor* specifying the desired RNA structural motif and a DNA sequence. The goal is to find all occurrences of the motif in the sequence. A descriptor consists of three parts:

1. a *motif map* – a list of individual *structural elements* ordered from 5' to 3' end along the sequence,
2. a detailed *specification* of each structural element,
3. an optional *search order*.

Each structural element is either single-stranded or paired (helical). *Single-stranded elements* are regular

expressions, similar to those used in PROSITE [35]. The user can also allow a fixed number of mismatches and insertions to appear anywhere in the motif. *Paired elements* correspond to helices in the RNA structure and consist of two interacting regions of the DNA sequence. The descriptor can specify the minimum and maximum length of the helix, sequence constraints in the form of a regular expression, as well as constraints on the paired bases (for example, we can consider only canonical Watson-Crick base-pairs, or allow U-G pairs as well). Again, users can allow a certain number of mispairs between paired bases, mismatches with respect to the sequence constraints, and insertions of single-base bulges. Each paired element occurs twice in the motif map, specifying the location of both strands. We place no restrictions on the relative order of elements in the motif map, and thus the descriptor can specify arbitrary pseudoknotted structures. An example of a descriptor is in Fig. 1, and the full description of the file format is given in Additional file 1: Section S1.

The user can optionally specify the search order in which individual elements will be considered in the backtracking search. The search order has a large influence on the running time, and the main focus of this paper is automated selection of appropriate search orders.

Algorithm outline The algorithm uses a simple backtracking strategy with a fixed search order of elements $e_1, e_2, \ldots, e_n$. First, we find all matches of element e_1 in a certain sequence window T. Then we consider each match of e_1 in turn and try to expand it to an occurrence of the complete motif by recursively searching for matches of $e_2, \ldots, e_n$ in appropriate relative positions with respect to the match of e_1. An illustration of the search procedure is depicted in Fig. 2.

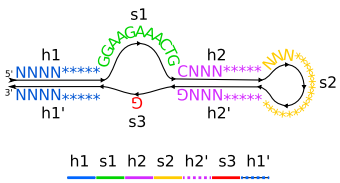
To find matches of element e_i, we devised general but relatively slow dynamic programming algorithms. For single-strand elements with no wild cards or insertions, we use a much faster bit-parallel bounded nondeterministic DAWG matching algorithm [36]. For the rest of the single-strand elements, we first use bit-parallel shift-and forward filtering [37] to identify sequence positions with possible element occurrences, and only subsequently we verify matches by the full dynamic programming algorithm.

The dynamic programming tables of our algorithms have many dimensions, because we need to keep track of the number of insertions, mismatches, and for paired elements also mispairs. In a typical sequence search scenario, each of these differences is assigned some negative score and the goal is to optimize the overall score. In contrast, we have a separate upper bound for each type of difference from the motif; therefore, each type adds another dimension to the dynamic programming table.

For example, for paired elements, our table H has seven dimensions. The value of $H_{t_1,t_2,p,m,r,i,b}$ is TRUE if and only if prefix $P[1 \ldots p]$ of the regular expression can be aligned with a suffix T' of $T[1 \ldots t_1]$ with m mismatches, a prefix $P'[1 \ldots p]$ of the regular expression for the reverse strand can be aligned with a prefix T'' of $T[t_2 \ldots |T|]$ with no mismatches, and the alignment of T' and T'' to each other contains i insertions and r mispairings. Furthermore, since we do not allow insertions to be adjacent, we use a binary flag b such that b is true if and only if one of $T[t_1]$ and $T[t_2]$ is an insertion. The complete dynamic programming recurrences can be found in Additional file 1: Section S2.

The number of allowed insertions, mismatches, and mispairs is typically very small, and thus the dynamic programming runs in $O(t^2k)$ time, where t is the length of the sequence window T and k is the length of the motif. The search procedure divides the whole sequence into windows of size $\max\{20L, 3000\}$, where L is the maximum length of an occurrence. Successive windows overlap by length L so that each occurrence is guaranteed to be completely contained in at least one window.

For the first element e_1 in the search order, we run the search on the whole window. For the successive elements, we compute a *search domain* in which this element may occur and restrict the dynamic programming accordingly. The search domain is determined based on the positions of the closest matches on the left and on the right already fixed in the previous steps of the backtracking search and by the flexibility in the length of the elements separating




```
h1  s1  h2  s2  h2'  s3  h1'

h1  0:0  NNNN*****:*****NNNN
h2  0:0  CNNN*****:*****NNNG

s1  0  GGAAGAAACTG
s2  0  NNN[17]
s3  0  G
```

Fig. 1 An illustration of an ATP aptamer motif and its corresponding descriptor based on genomic adenosine aptamers [21]. Nucleotide constraints for individual positions are expressed in the IUPAC notation [51]

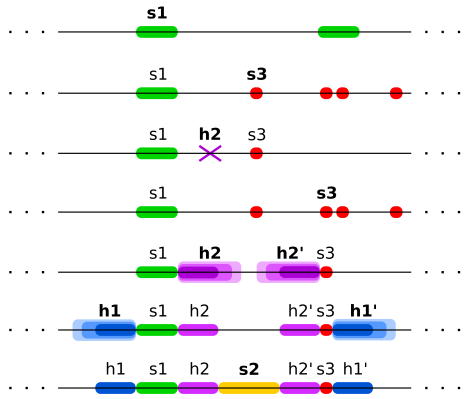


Fig. 2 An illustration of the RNArobo search procedure for the motif of ATP aptamer. The search follows the order of elements `s1`, `s3`, `h2`, `h1`, `s2`

the matches of previously fixed elements from the current element, as illustrated in Fig. 3.

The overall running time of our algorithm can be, in the worst case, exponential in the number of elements of the descriptor. However, the number of these elements is typically small, and if we use a well-chosen search order, the early branching elements will have relatively few matches, thus limiting the degree of the search tree. Many branches of the search are terminated early, because no match of an element is found in its search domain.

Element ordering

The search order of elements significantly affects the running time. In this section, we present our data-driven element ordering (DDEO) strategy. In general, it is advantageous to start with elements that have few matches, thus eliminating a large portion of the sequence to be searched. Once the matches of some elements are found, it is also important to consider flexibility of the placement of a new element with respect to those that are already matched.

While these principles are quite natural, it is difficult to transform them into an effective criterion for creating good search orders. Therefore, we propose a data-driven method for finding a close-to-optimal element ordering.

Our approach consists of two parts. First, we use a heuristic approach to create a set O of candidate orderings. We then use these orderings on sequence windows, measure their actual performance, and select the best one. We do this while processing the initial segment of the query sequence, thus limiting the amount of overhead spent on selecting a suitable search order.

Heuristic proposal of element orders

To create the proposal set O, we concentrate on the first k-tuple of elements in the search order (in experiments, we use $k = 3$). We create all possible k-tuples and score them by the heuristic scoring function described below. All the k-tuples with scores above some threshold are then augmented to complete search orders forming the proposal set O. In experiments we select k-tuples that achieve score that is at least 85 % of the maximum among all k-tuple scores. We limit the size of O to 50 if there are too many good candidates.

The goal of this initial heuristic evaluation is to select a small subset of k-tuples to be evaluated empirically. We want this subset to include k-tuples that can be augmented to complete search orders yielding running times close to the optimum. Conversely, we should not include too many tuples yielding slow running times, because their evaluation will increase the overall running time.

The score of a k-tuple $e_1, \ldots, e_k$ is a weighted sum of two heuristic functions evaluated for individual elements

$$h(e_1, \ldots, e_k) = \sum_{i=1}^{k} 2^{k-i} \left(c_1 \cdot h_1(e_i) + c_2 \cdot h_2(e_1, \ldots, e_i)\right) \tag{1}$$

Function $h_1(e_i)$ approximates the information content of element e_i and function $h_2(e_1, \ldots, e_i)$ considers flexibility of element e_i with respect to the already matched elements $e_1, \ldots, e_{i-1}$. Note that element weights decrease exponentially, because elements placed earlier in the

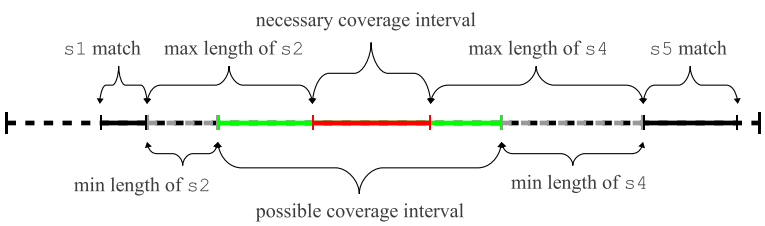


Fig. 3 Computation of the search domain for a single-stranded element `s3`. Here we have a partly matched motif composed of five single-stranded elements `s1`, . . . ,`s5`. Assume that elements `s1` and `s5` have already been matched. The match of `s3` has to start in the left green interval and end in the right green interval, and it has to completely cover the red interval in the middle

search order tend to be searched in a larger portion of the sequence. We set the weight c_2 in the linear function to -0.2, while c_1 is 1 for paired elements and 3 for single-strand elements to reflect that the search for unpaired elements is considerably faster.

Information content heuristic

The first heuristic function h_1 is an approximation of the information content of an element, favoring elements that pose more specific constraints. Thus this function follows the *fail-first* rule generally used in backtracking searches [38]. *Information content* is a measure of uncertainty reduction about an outcome once we have received a new piece of information. In particular, it is the difference in the entropy of a random variable before and after some message has been received. The entropy of a discrete random variable X with possible values $x_1, x_2, \ldots$ is defined as

$$H(X) = -\sum_i P[X = x_i] \log_2 P[X = x_i]. \quad (2)$$

Let us first consider a single-stranded element S, and let N be the longest possible occurrence of this element. In our setting, the random variable is a sequence of length N and the message is that the sequence starts with a match of the pattern. To estimate the background entropy before receiving the message, we consider all 4^N sequences of length N equally likely, obtaining

$$H_{\text{before}} = -\sum_{i=1}^{4^N} \frac{1}{4^N} \log_2 \frac{1}{4^N} = 2N. \quad (3)$$

We compare this value with the entropy of the uniform distribution over all sequences of length N that have an occurrence of the element starting at the first position. If X is the number of such sequences, we have $H_{\text{after}} = \log_2 X$ and the information content of S is

$$h_1(S) = H_{\text{before}} - H_{\text{after}} = 2N - \log_2 X. \quad (4)$$

Since the value of X is hard to compute for complex elements, we use an upper bound $X_U \geq X$ (which leads to a lower bound for the information content of S). To obtain the upper bound X_U, we count different ways of obtaining a sequence matching S, disregarding the fact that some sequences may be obtained in several different ways and consequently counted multiple times.

In the simplest case, element S does not contain any flexible-length wild cards and does not allow for any distortions (mismatches, insertions). The element specifies for each position i the set of allowed nucleotides; let $C[i]$ be the size of this set. The value of X is then simply

$$X = \prod_{i=1}^{N} C[i]. \quad (5)$$

Next we extend the bound to cases when S contains wild cards. Each wild card corresponds to an arbitrary nucleotide or to an empty string. A block of k consecutive wild cards thus corresponds to an arbitrary sequence of length up to k. Let X_1 be the value obtained by formula (5) if we consider only non-wild card positions in S. A single block of k consecutive wild cards increases the value of N (the length of the longest occurrence of S) by k. These k additional nucleotides can be arbitrary, and are split into a block of length i matching the block of wild cards and a block of length $k - i$ located after the element occurrence (this block corresponds to the unused wild cards). Since the value of i can be any integer between 0 and k, this leads to the upper bound of $X_1(k+1)4^k$ sequences matching S. If S has multiple blocks of wild cards of lengths $k_1, \ldots, k_b$, each of them can be split into two blocks independently, leading to the upper bound

$$X_2 = X_1 \cdot \prod_{i=1}^{b} 4^{k_i}(k_i + 1). \quad (6)$$

Similarly, we adjust the value of X to account for mismatches and insertions allowed in the element to obtain the final upper bound X_U; see details in Additional file 1: Section S3. For practical reasons, we handle mismatches using a formula which is not guaranteed to be an upper bound of the real set size X for each motif, but works well in practice.

The situation is analogous for paired elements. Let H be an element consisting of two paired strands H_1 and H_2, and let N be the maximum length of a match to one of these two strands, after accounting for wild cards and insertions. Since we now consider sequences of total length $2N$, the background entropy is

$$H_{\text{before}} = \log_2 4^{2N} = 4N.$$

We use $H_{\text{after}} = \log_2 X$, where X is the number of pairs of sequences of length N such that H_1 occurs in the first sequence starting at the first position, and H_2 occurs in the second sequence ending at the last position, and these two occurrences satisfy the complementarity constraints with up to allowed amount of distortion. We again use an approximate upper bound X_U instead of the actual count X, counting different ways that such a matching can occur.

As with single-stranded elements, we first count the number of sequences that match H without considering wild cards and distortions. Let $P[i]$ be the number of valid base pairs between position i of H_1 and the corresponding position of H_2. The value of $P[i]$ is determined by both complementarity constraints specified by H and by

sequence constraints for the respective positions in H_1 and H_2. As before, the number of matching sequences is the product

$$X_1 = \prod_{i=1}^{N} P[\,i\,]\,. \tag{7}$$

To obtain the final bound X_U, we adjust the value of X_1 to account for wild cards, mismatches, and insertions, similarly as in the single-stranded case. We also adjust the bound for allowed mispairs, where the two paired nucleotides do not form a valid base pair. Details can be found in Additional file 1: Section S3.

Domain flexibility heuristic

The second heuristic scoring function $h_2(e_i)$ measures the flexibility of positioning element e_i with respect to already matched elements $e_1, \ldots, e_{i-1}$. The matches of these elements specify the search domain for element e_i, as shown in Fig. 3. Longer domains require more time for finding matches and are also likely to yield more matches, each of which will be then examined individually in backtracking. Therefore, we set the weight c in (1) to be negative.

To compute the exact size of the search domain, we need to know positions of matches of $e_1, \ldots, e_{i-1}$. In order to score a particular search order before the search starts, we need to approximate flexibility of e_i without this knowledge. For an unpaired element e_i, we find the nearest fixed element e_ℓ on the left side of e_i (one of $e_1, \ldots, e_{i-1}$). Then we sum up the flexibilities of all elements between e_ℓ and e_i in the descriptor, where the flexibility of an element is the difference between its maximum and minimum length. We denote this sum F_{left}. Analogously, we obtain F_{right} for the right side of e_i. If there is a fixed element on both sides of e_i, we take the minimum of F_{left} and F_{right}, as both are upper bounds on the search domain size for e_i:

$$h_2(e_1, \ldots, e_i) = \min\{F_{\text{left}}, F_{\text{right}}\}. \tag{8}$$

If there is no fixed element at one side, we only use the other side to compute h_2. For a paired element e_i, we first compute flexibilities of the two strands individually, while considering the other strand to be fixed. Then we take the maximum of these two.

Candidate order elimination

The heuristic function defined above is used to select promising initial k-tuples for search orders. These are then completed to full search orders by adding remaining elements in the order determined by the information content heuristic, forming the initial candidate set O.

The initial candidate set contains a mix of good and bad orderings. We process several window s of the sequence to determine which orderings are good. For each window, we sample a random search order x from O and use it in the search procedure, measuring its performance T_x. In particular, we record nanoseconds used by the process (measured by an available system function) and normalize it by the window size. Based on the gathered data, we continually eliminate orderings with bad performance using a statistical test.

We treat T_x as a random variable, and approximate it by the normal distribution with an unknown mean and variance. Our goal is to pick from the set of candidate orderings O the ordering leading to the shortest mean execution time. Formally, we want to find $x^* \in O$, such that for each y, $E[\,T_{x^*}] \leq E[\,T_y]$. Given several observed values of T_x and T_y, we use Welch's t-test [39] to test the null hypothesis $E[\,T_x] \geq E[\,T_y]$ against the alternative hypothesis $E[\,T_x] < E[\,T_y]$. This test is used for hypothesis testing concerning difference between the actual means of two normally distributed populations with possibly unequal variances, based on independent sets of samples from these distributions [40].

Each time we gather a new sample from T_x for some $x \in O$, we test x against the rest of O. When we observe a statistically significant difference between two candidates (at the level $\alpha = 0.01$), we eliminate the one with the higher mean time of execution from set O. If two candidates cannot be shown significantly different, even after both were sampled many times (75 samples from each), we simply eliminate the ordering with the higher sample mean.

Once we eliminate all but one search order from set O, we start to refine this final ordering. Recall that it consists of an initial k-tuple extended to a full ordering by the information content heuristic. We drop this heuristic extension, and start training the following k-tuple in the search order with the first k-tuple already fixed. We continue until we completely fix the search order.

Performance of DDEO

We have evaluated DDEO heuristic on the hepatitis delta virus like ribozyme (HDV) descriptor (Fig. 4). Even though the entropy-based heuristic is not perfect and

```
h1 s1 h3 r4 s2 r4' h1' s4 h4 s5 h4' s7 h3' s8
h1 0:0 GNNNNN*:*NNNNNY
s1 0   N[50]
h3 0:0 NNNNNN**:**NNNNNN
r4 0:0 NNN:NNN TGCA
s2 0   TYYHCG*Y
s4 0   RN
h4 0:0 NNNNN*:*NNNNN
s5 0   NNN***
s7 0   CNRA*
s8 0   NNNNNN
```

Fig. 4 Descriptor for HDV-like ribozyme with structured P4 region. The motif contains four paired elements organized in a double pseudoknot

the candidate set contains bad orderings in addition to the good ones, the orderings with bad performance are eliminated after only a few samples (in some cases as few as two). These "bad runs" thus do not increase the overall running time significantly. (More details can be found in Additional file 1: Section S4).

Results and Discussion

We have performed several experiments comparing RNArobo with other established RNA motif search tools: RNAbob ([5], software version 2.2.1 from 2012), RNAmotif ([8], software version 3.0.7 from 2010), RNAMot ([41], software version 2.1 from 1994), and RaligNAtor [9]. We have concentrated on both the speed and the ability to discover biologically meaningful motif occurrences.

Accuracy of hits Table 1 shows the results of an experiment, where we revisited several scientific studies involving discovery of ribozymes and aptamers [7, 16, 42]. We have constructed RNA descriptors for RNAbob and used both RNAbob and our new tool RNArobo to identify motif occurrences. Extended description of the experiment and the descriptors are included in Additional file 1: Section S5. As expected, the results of the two programs were identical, with RNArobo running much faster (data not shown, but see speed evaluation below). Almost all of the hits found by the programs were occurrences of known targets also identified in the original studies, confirming the accuracy of the algorithm (see exceptions below).

Novel findings Compared to RNAbob, RNArobo allows insertions (bulges) in helix elements, enabling much more flexible descriptors better characterizing certain families. For example, in the case of hammerhead type I ribozyme (HHR type I, 4 bp), the descriptor with insertions identified 15 known occurrences in *Yarrowia lipolytica* ("Yli" family) [16], compared to a single occurrence identified without insertions.

Allowing insertions, we have also discovered several new candidate occurrences. Firstly, three new hits of hammerhead ribozyme (HHR type II) in *B. cereus* are likely false positives as determined by the Fold-Filter pipeline [43]. This pipeline uses tools from ViennaRNA package [44] and DotKnot [45] to determine if an occurrence of a motif is likely to assume the secondary structure implied by the descriptor.

On the other hand, a novel GTP aptamer (GTP class I) in Davis and Szostak library is likely functional, since the library was selected for GTP binding.

In case of hammerhead ribozyme (HHR type I, 3 bp) in *Y. lipolytica*, the number of hits increased massively from 4 to 54 by allowing insertions. These hits form two distinct families. The first contains previously known "Yli" ribozymes, as identified by Perreault et al. [16]. However, the ten hits of the second family are novel and pass through the Fold-Filter pipeline. They likely represent a novel HHR family in *Y. lipolytica* genome similar to a large family of HHRs in the *Schistosoma mansoni* genome (see also Additional file 1: Figure S7); no HHR type I families besides the "Yli" ribozymes were previously found in *Y. lipolytica*.

Table 1 Summary of reproducing results from the literature using RNArobo. Searches were extended by allowing for insertions in structurally conserved elements that are known to tolerate single base insertions. This extension led to improved sensitivity and yielded several new putatively functional ribozymes

Sequence	*Descriptor*	*#hits*	*Known targets found*	*Note*
In vitro selected library	GTP apt. class I	9	yes	
(∼65 kBp scanned)	GTP apt. class I w/ ins	10	yes [42]	novel hit
Yarrowia lipolytica	HHR I (4 bp)	1	Yli-1-3	
(∼41 MBp scanned)	HHR I (4 bp) w/ ins	15	Yli-1-3 through Yli-1-11	
	HHR I (3 bp)	4	Yli-1-3 and Yli-1-13	
	HHR I (3 bp) w/ ins	54	Yli-1-3 through Yli-1-11, and Yli-1-13 [16]	novel family (10 hits)
Bacillus cereus	HHR II	1	Bce-1-1	
(∼11 MBp scanned)	HHR II w/ ins	4	Bce-1-1 [16]	
Anopheles gambiae chr2L	HDV (loose P4)	7	Agam-1-1	
(∼98 MBp bases scanned)	HDV (loose P4) w/ ins	36	Agam-1-1 and Agam-1-2 [7]	
Strongylocentrotus purpuratus	HDV (stem P4)	11	yes	
(∼2.1 GBp scanned)	HDV (stem P4) w/ ins	11	yes	
	HDV (loose P4) + FF	15	yes	
	HDV (loose P4) w/ ins + FF	16	yes [7]	novel hit

FF: only hits passing the Fold-Filter are reported

Allowing insertions in HDV-like ribozymes in *S. purpuratus* genome did not yield any new hits (HDV stem P4 descriptor). We have attempted to loosen the constraints on the P4 region, which yielded numerous hits both with and without insertions. Consequently, we have applied Fold-Filter and kept only hits passing the pipeline. The descriptor with insertions yielded a previously unidentified hit that aligns well with other HDV ribozymes, thus likely being functional. The above examples show that RNArobo can be helpful in finding interesting novel occurrences, even for known families established in the literature.

Speed comparison Table 2 shows the comparison of running times of scanning both strands of the whole human genome for occurrences of nine realistic RNA motifs. In most cases, RNArobo is the fastest tool, in many cases speeding up the search more than 100-fold. Note that running times of both RNAbob and RNAmotif greatly depend on the complexity of the motif, while RNArobo does not show pronounced dependency on the descriptor. This is most apparent for the HDV descriptors which feature a double pseudoknot. Allowing insertions in the helices does not significantly slow down RNArobo search.

RaligNAtor [9] is a recent addition to the family of descriptor-based search tools. In contrast to the previous works, the authors add a preprocessing step building index data structures that help to speed up the subsequent searches. Unfortunately, the structural pattern definition language of RaligNAtor is very different from that used by other tools; therefore it is difficult to translate RNArobo patterns to RaligNAtor and vice versa. Nevertheless, we defined approximate counterparts of two patterns (IRES from RaligNAtor tests and generalized tRNA from our tests) in the other descriptor language in order to compare the two tools. Results convincingly show that RNArobo outperforms RaligNAtor in terms of the running time from four-fold to more than 1000-fold (Additional file 1: Table S2). These results hold whether we used exact or approximate patterns, with or without preprocessing (see Additional file 1: Section S7).

Comparison to convariance-model-based tools Descriptor-based tools, such as RNArobo, are used by researchers to explore motif families for which only a few occurrences are known, and are heavily based on incorporating user's intuition in building motif descritors. On the other hand, covariance models, such as InfeRNAl [3], can be used to search for new instances for motif families where many occurrences are already known, and their common features are extracted automatically and encoded in the covariance model. Due to this substantial difference, it is difficult to design a fair comparison between these two classes of algorithms.

Nevertheless, we have attempted to compare RNArobo to InfeRNAl in case of the hammerhead ribozyme family that has been previously extensively studied by several groups. InfeRNAl was generally 3–10 times slower than RNArobo when searching for these motifs in *Yarrowia lipolytica*. Both programs identify bona fide ribozymes; however, InfeRNAl also identifies candidates with mutations in the active site of the catalytic RNA, making these instances most likely inactive. Such results may be useful for RNAs that tolerate a certain level of global mutation rate, but in our case we most likely find false positives. The descriptor of RNArobo allows specification of regions that are under strong purifying selection and need to be conserved; on the other hand, allowing insertions in helices allowed RNArobo to discover a new putative family of *Yarrowia lipolytica* hammerhead ribozymes that InfeRNAl did not find.

Conclusions

In this work, we have developed a new tool RNArobo for RNA motif search. RNArobo allows expert human users to describe the most relevant features of a target RNA structure and then to search for distant homologs in available genomic data. The focus of our work was an automated strategy for element ordering in the backtracking search employing a heuristic scoring function based on information content and search domain size estimates. We used statistical tests to eliminate candidate orderings that do not perform well in practice. Our experiments

Table 2 The running times (in seconds) of different programs searching for various descriptors in the whole human genome

	ATP apt.	GTP apt. class I	generalized tRNA	HHR-I (4 bp)	HHR-II (3 bp)	HHR extended	HDV (loose P4)	HDV (stem P4)	HDV (mispairs)
RNAbob	3,419.03	2,744.30	7,450.07	923.53	5,027.33	2,269.35	209,932.57	43,430.78	36,459.87
RNAmotif	**80.78**	222.54	7,374.32	**87.69**	265.09	116.15	26,259.79	2,513.90	9,240.83
RNAMot	unf	unf	unf	unf	unf	unf	unf	4,538.92	8,925.31
RNArobo	80.91	**151.09**	**250.63**	96.48	**110.64**	**108.46**	**171.16**	**169.90**	**111.30**
RNArobo-ins	–	153.38	–	98.22	137.47	–	173.82	171.54	–

Experiments were run on Intel Xeon E5520 CPU. RNArobo-ins is RNArobo run with modified descriptors allowing insertions in helical elements. RNAMot did not finish on most of the inputs within time limit of three days. Only results that finished within three days are shown. Since DDEO is randomized, we show the average running time of five runs of RNArobo. Standard deviation was up to 3 % or 5 sec, with the exception of the HHR extended descriptor, where the running time ranged from 98 to 125 sec
Boldface numbers represent the best running times for a particular descriptor

demonstrate that RNArobo is much faster for complex motifs than existing tools, thus facilitating large-scale whole-genome searches.

Our work leaves open further avenues for research. The problem of finding the best element ordering, or even estimating the expected number of matches of a single element in a random sequence, is very intriguing from the theoretical point of view. Even though our simple elimination scheme proved to be effective in practice, it would be interesting to treat the problem as an on-line learning problem and develop a theory that would allow us to estimate how fast a particular elimination algorithm converges to the best (or close to the best) ordering. Our heuristic scoring function can perhaps be improved by adding more partial scores and combining them with weights estimated by regression techniques from performance data observed on several descriptors. Finally, DNA sequences are non-uniform, and a scheme that could adapt to changing character of sequences as they are processed would likely lead to further improvement of our algorithm. An interesting application of our algorithm would be assigning basic structural motifs to sequences as they are produced by high-throughput sequencers.

A logical extension of our problem is to construct descriptors automatically from known examples of RNAs. A step in this direction has already been taken and several algorithms to locate common substructures of two RNAs were developed [46, 47]. Such patterns are then basis of several practical tools for pattern-based RNA comparison, including ExpaRNA [48], LocARNA [49], and ExpaRNA-P [50].

Ethics approval and consent to participate
not applicable.

Consent for publication
not applicable.

Availability of data and material
Software is available at http://compbio.fmph.uniba.sk/rnarobo. Data sets are publicly available as outlined in the paper. Descriptors for the searches are provided in Additional file 1.

Additional file

Additional file 1: Supplementary online material. The file contains supplementary material with additional details on methods, file formats, and experiments. (PDF 617 kb)

Abbreviations
not applicable.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
AL, TV, and BB conceived the study. LR, TV, and BB developed algorithms and methods. LR implemented the software. LR, and RJ designed and executed the experiments. All authors have participated on writing the article. All authors read and approved the final manuscript.

Acknowledgements
not applicable.

Funding
This research was funded by VEGA grants 1/0684/16 (BB) and 1/0719/14 (TV), Slovak Research and Development Agency grant APVV-14-0253, Pew Charitable Trusts (AL), NIH EUREKA 5R01GM094929 (AL), NSF MCB 1330606 (AL), NIH 1F31GM103241 (RMJ).

Authors' information
not applicable.

Author details
[1] Department of Computer Science, University of Toronto, Toronto, ON M5R 3G4, Canada. [2] Department of Pharmaceutical Sciences, Chemistry, and Molecular Biology and Biochemistry, University of California, Irvine, 2141 Natural Sciences 2, Irvine, CA, 92697, USA. [3] Faculty of Mathematics, Physics, and Informatics, Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia.

Received: 29 September 2015 Accepted: 7 May 2016
Published online: 18 May 2016

References

1. Griffiths-Jones S, Bateman A, Marshall M, Khanna A, Eddy SR. Rfam: an RNA family database. Nucleic Acids Res. 2003;31(1):439–41.
2. Yao Z, Weinberg Z, Ruzzo WL. CMfinder–a covariance model based RNA motif finding algorithm. Bioinformatics. 2006;22(4):445–52.
3. Nawrocki EP, Kolbe DL, Eddy SR. Infernal 1.0: inference of RNA alignments. Bioinformatics. 2009;25(10):1335–7.
4. Gautheret D, Major F, Cedergren R. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. Comput Appl Biosci. 1990;6(4):325–1.
5. Eddy SR. RNABob: a program to search for RNA secondary structure motifs in sequence databases. 1996. unpublished.
6. Reeder J, Reeder J, Giegerich R. Locomotif: from graphical motif description to RNA motif search. Bioinformatics. 2007;23(13):392–400.
7. Webb CH, Riccitelli NJ, Ruminski DJ, Luptak A. Widespread occurrence of self-cleaving ribozymes. Science. 2009;326(5955):953.
8. Macke TJ, Ecker DJ, Gutell RR, Gautheret D, Case DA, Sampath R. RNAMotif, an RNA secondary structure definition and search algorithm. Nucleic Acids Res. 2001;29(22):4724–35.
9. Meyer F, Kurtz S, Beckstette M. Fast online and index-based algorithms for approximate search of rna sequence-structure patterns. BMC Bioinforma. 2013;14(1):226.
10. Nawrocki EP, Burge SW, Bateman A, Daub J, Eberhardt RY, Eddy SR, Floden EW, Gardner PP, Jones TA, Tate J, Finn RD. Rfam 12.0: updates to the RNA families database. Nucleic Acids Res. 2015;43(Database issue):130–7.
11. Ferbeyre G, Smith JM, Cedergren R. Schistosome satellite DNA encodes active hammerhead ribozymes. Mol Cell Biol. 1998;18(7):3880–8.
12. Rojas AA, Vazquez-Tello A, Ferbeyre G, Venanzetti F, Bachmann L, Paquin B, Sbordoni V, Cedergren R. Hammerhead-mediated processing of satellite pDo500 family transcripts from Dolichopoda cave crickets. Nucleic Acids Res. 2000;28(20):4037–3.
13. Martick M, Horan LH, Noller HF, Scott WG. A discontinuous hammerhead ribozyme embedded in a mammalian messenger RNA. Nature. 2008;454(7206):899–902.
14. Przybilski R, Graf S, Lescoute A, Nellen W, Westhof E, Steger G, Hammann C. Functional hammerhead ribozymes naturally encoded in the genome of Arabidopsis thaliana. Plant Cell. 2005;17(7):1877–85.
15. Jimenez RM, Delwart E, Luptak A. Structure-based search reveals hammerhead ribozymes in the human microbiome. J Biol Chem. 2011;286(10):7737–43.

16. Perreault J, Weinberg Z, Roth A, Popescu O, Chartrand P, Ferbeyre G, Breaker RR. Identification of hammerhead ribozymes in all domains of life reveals novel structural variations. PLoS Comput Biol. 2011;7(5):1002031.
17. Seehafer C, Kalweit A, Steger G, Graf S, Hammann C. From alpaca to zebrafish: hammerhead ribozymes wherever you look. RNA. 2011;17(1): 21–6.
18. Webb C-HT, Lupták A. HDV-like self-cleaving ribozymes. RNA Biol. 2011;8(5):719–27.
19. Ruminski DJ, Webb C-HT, Riccitelli NJ, Lupták A. Processing and translation initiation of non-long terminal repeat retrotransposons by hepatitis delta virus (HDV)-like self-cleaving ribozymes. J Biol Chem. 2011;286(48):41286–95.
20. Riccitelli NJ, Delwart E, Luptak A. Identification of minimal HDV-like ribozymes with unique divalent metal ion dependence in the human microbiome. Biochemistry. 2014;53(10):1616–1616.
21. Vu MMK, Jameson NE, Masuda SJ, Lin D, Larralde-Ridaura R, Luptak A. Convergent evolution of adenosine aptamers spanning bacterial, human, and random sequences revealed by structure-based bioinformatics and genomic SELEX. Chem Biol. 2012;19(10):1247–54.
22. Rampášek L. RNA structural motif search is NP-complete. In: Proceedings of the Student Science Conference 2011, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava; 2011. p. 341–8. http://compbio.fmph.uniba.sk/svk2011/svk2011-zbornik.pdf.
23. Lathrop RH. The protein threading problem with sequence amino acid interaction preferences is NP-complete. Protein Eng Des Sel. 1994;7(9): 1059.
24. Jiang T, Lin G, Ma B, Zhang K. A general edit distance between RNA structures. J Comput Biol. 2002;9(2):371–88.
25. Rinaudo P, Ponty Y, Barth D, Denise A. Tree decomposition and parameterized algorithms for RNA structure-sequence alignment including tertiary interactions and pseudoknots. In: Algorithms in Bioinformatics (WABI). Lecture Notes in Computer Science, vol. 7534. Heidelberg: Springer; 2012. p. 149–64.
26. Billoud B, Kontic M, Viari A. Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. Nucleic Acids Res. 1996;24(8):1395.
27. Grillo G, Licciulli F, Liuni S, Sbisa E, Pesole G. PatSearch: a program for the detection of patterns and structural motifs in nucleotide sequences. Nucleic Acids Res. 2003;31(13):3608.
28. Chang TH, Huang HD, Chuang TN, Shien DM, Horng JT. RNAMST: efficient and flexible approach for identifying RNA structural homologs. Nucleic Acids Res. 2006;34:423–8.
29. George AD, Tenenbaum SA. Informatic resources for identifying and annotating structural RNA motifs. Mol Biotechnol. 2009;41(2):180–93.
30. Meyer F, Kurtz S, Backofen R, Will S, Beckstette M. Structator: fast index-based search for RNA sequence-structure patterns. BMC Bioinforma. 2011;12:214.
31. Drory Retwitzer M, Polishchuk M, Churkin E, Kifer I, Yakhini Z, Barash D. RNAPattMatch: a web server for RNA sequence/structure motif detection based on pattern matching with flexible gaps. Nucleic Acids Res. 2015;43(W1):507–12.
32. Strothmann D. The affix array data structure and its applications to rna secondary structure analysis. Theor Comput Sci. 2007;389(1):278–94.
33. Abouelhoda MI, Kurtz S, Ohlebusch E. Replacing suffix trees with enhanced suffix arrays. J Discret Algorithm. 2004;2(1):53–86.
34. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990;215(3):403–10.
35. Bairoch A. PROSITE: a dictionary of sites and patterns in proteins. Nucleic Acids Res. 1991;19 Suppl:2241–5.
36. Navarro G, Raffinot M. Fast and flexible string matching by combining bit-parallelism and suffix automata. J Exp Algorithmic (JEA). 2000;5:4.
37. Navarro G, Raffinot M. Fast and simple character classes and bounded gaps pattern matching, with applications to protein searching. J Comput Biol. 2003;10(6):903–23.
38. Russell SJ, Norvig P. Artificial Intelligence: A Modern Approach. Upper Sadle River: Prentice Hall; 2010.
39. Welch BL. The generalization of 'Student's' problem when several different population variances are involved. Biometrika. 1947;34(1/2):28–35.
40. Ruxton GD. The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test. Behav Ecol. 2006;17(4): 688–90.
41. Laferrière A, Gautheret D, Cedergren R. An RNA pattern matching program with enhanced performance and portability. Comput Appl Biosci. 1994;10(2):211–2.
42. Davis JH, Szostak JW. Isolation of high-affinity GTP aptamers from partially structured RNA libraries. Proc Natl Acad Sci. 2002;99(18):11616–21.
43. Jimenez RM, Rampášek L, Brejová B, Vinař T, Lupták A. Discovery of RNA motifs using a computational pipeline that allows insertions in paired regions and filtering of candidate sequences. Methods Mol Biol (Clifton, NJ). 2012;848:145.
44. Lorenz R, Bernhart SH, zu Siederdissen CH, Tafer H, Flamm C, Stadler PF, Hofacker IL. ViennaRNA package 2.0. Algorithm Mol Biol. 2011;6(1):26.
45. Sperschneider J, Datta A. DotKnot: pseudoknot prediction using the probability dot plot under a refined energy model. Nucleic Acids Res. 2010;38(7):103–3.
46. Backofen R, Siebert S. Fast detection of common sequence structure patterns in rnas. J Discret Algorithm. 2007;5(2):212–28.
47. Amit M, Backofen R, Heyne S, Landau GM, Mohl M, Otto C, Will S. Local Exact Pattern Matching for Non-Fixed RNA Structures. IEEE/ACM Trans Comput Biol Bioinform. 2014;11(1):219–20.
48. Heyne S, Will S, Beckstette M, Backofen R. Lightweight comparison of RNAs based on exact sequence-structure matches. Bioinformatics. 2009;25(16):2095–102.
49. Will S, Reiche K, Hofacker IL, Stadler PF, Backofen R. Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. PLoS Comput Biol. 2007;3(4):65.
50. Otto C, Mohl M, Heyne S, Amit M, Landau GM, Backofen R, Will S. ExpaRNA-P: simultaneous exact pattern matching and folding of RNAs. BMC Bioinforma. 2014;15:404.
51. Cornish-Bowden A. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. Nucleic Acids Res. 1985;13(9):3021.