JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 14, Number 1, 2007

Pp. 33–44
DOI: 10.1089/cmb.2006/0089

A Dynamic Programming Approach for Finding Common Patterns in RNAs

SVEN SIEBERT and ROLF BACKOFEN

ABSTRACT

We developed a dynamic programming approach of computing common sequence structure patterns among two RNAs given their primary sequences and their secondary structures. Common patterns between two RNAs are defined to share the same local sequential and structural properties. The locality is based on the connections of nucleotides given by their phosphodiester and hydrogen bonds. The idea of interpreting secondary structures as chains of structure elements leads us to develop an efficient dynamic programming approach in time $O(nm)$ and space $O(nm)$, where n and m are the lengths of the RNAs. The biological motivation is given by detecting common, local regions of RNAs, although they do not necessarily share global sequential and structural properties. This might happen if RNAs fold into different structures but share a lot of local, stable regions. Here, we illustrate our algorithm on Hepatitis C virus internal ribosome entry sites. Our method is useful for detecting and describing local motifs as well. An implementation in C++ is available and can be obtained by contacting one of the authors.

Key words: local patterns, similar RNAs, related RNAs, sequence structure, RNA motifs.

1. INTRODUCTION

RNAs ARE POLYMERS consisting of the four nucleotides A, C, G, and U, which are linked together by their phosphodiester bonds. This chain of nucleotides is called the primary sequence. Bases which are part of nucleotides form hydrogen bonds within the same molecule leading to structure formation. One major challenge is to find (nearly) common patterns in RNAs since they suggest functional similarities of these molecules. Research on sequential and structural features are associated with considerable efforts. The structure in combination with the sequence of a molecule dictates its function. For instance, the presence of a SECIS (Selenocysteine Insertion Sequence) element in the $3'$ region of a UGA stop codon is known to inhibit the termination of the translation in all kingdoms of life. We will give an example of SECIS elements in *Methanococcus janaschii* at the end of this paper. Due to their functional similarities of related RNAs, RNA sequence structure analysis became so popular in recent years.

Our aim is to detect common local sequence structure patterns among two RNAs given their primary sequences and their secondary structures. We propose an algorithm which can list all patterns between two

Department of Bioinformatics, Institute of Computer Science, Albert-Ludwigs-University Freiburg, Freiburg, Germany.

RNAs in time $O(nm)$ and space $O(nm)$, where n and m are the lengths of the RNAs, respectively. The patterns are maximally extended—i.e., each of them has the largest size such that no pattern includes the maximally extended one. The maximal extensibility condition avoids falling into the trap of NP-hardness. For the patterns it means that, first, the output of proper sub-patterns is omitted, and, second, if there are at least two possibilities to build a maximally extended pattern, then these possibilities are split into a maximally extended and into a disjoint pattern. An example of the latter case follows.

The key idea of our dynamic programming method is to describe secondary structures not only as base-pairing interactions but also as structure elements known as hairpin loops, stacks (short form for stacked base-pairs), right bulges, left bulges, internal loops or multi-branched loops (Fig. 1). By performing computations on these loop regions from inside to outside, we obtain an elegant solution to the pattern finding problem. This step is made possible because base-pairs which enclose loops occur in a nested fashion, i.e., nested base-pairs fulfill for any two base-pairs (i_1, i_2) and (j_1, j_2) either $i_1 < i_2 < j_1 < j_2$ or $i_1 < j_1 < j_2 < i_2$.

A naive attempt is to consider all combinations of positions i in the first RNA and positions j in the second RNA and to extend these starting patterns by taking into account all neighboring nucleotides in both RNAs. The neighborhood of a nucleotide is defined as the set of nucleotides which are immediately connected through their phosphodiester or hydrogen bonds. If the considered nucleotides in the neighborhood of each RNA share the same sequential and structural properties then they are taken into the pattern, and the size of this pattern is increased by one. At first glance, this idea may work, but the crucial point are the loops. Consider, for example, the case depicted in Figure 2: suppose the algorithm starts at position 1 (lower left corner) in the first RNA and position 1 in the second RNA and is working towards the multi-branched loop in the first RNA. The lower stem has been successfully matched. But now there is no clear decision on whether to match the upper part of the stem-loop of the second RNA to the left side or to the right side of the

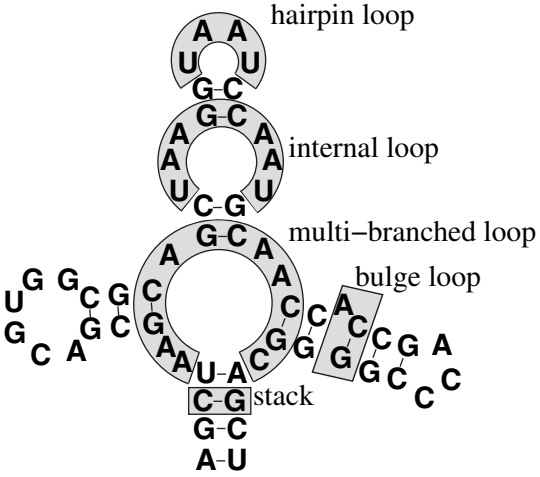

FIG . 1. Structure elements of an RNA secondary structure.

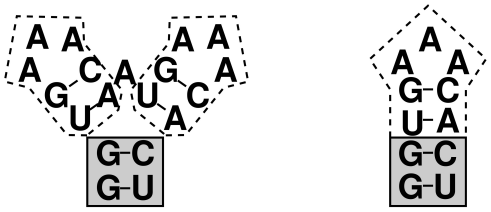

FIG . 2. Example of extending a common pattern among two RNAs. There is no clear decision on whether to match the upper part of the stem-loop of the right RNA to the left side or to the right side of the multi-branched loop in the left RNA.

multi-branched loop. This decision depends on how a common pattern is defined, of course, and on how to reach the maximally extended pattern. It is clear that the only solution here is to make some pre-computations on sequential and structural components of RNAs. Finally, we obtain a dynamic programming algorithm which compares inner parts of RNAs first, stores the results in various matrices and builds up the solutions successively. Note, that it is also a mistake to find common sequential parts first and then to recompose them by their structural properties. This is obviously a computationally intractable problem since all combinations of subsets of sequence parts have to be considered.

We illustrate our algorithm on Hepatitis C virus internal ribosome entry sites. They are folded into different minimum free energy structures, but share a lot of local sequence structure properties as our program ascertains. The detection of common patterns between two RNAs can be used as a global alignment. A selection of non-overlapping patterns can be arranged in such a way that a global alignment consists of exact sequence structure parts disrupted by non-exact parts. The main application of this algorithm here is to detect large local sequence structure parts in order to find interesting regions of biologically functional similarities. Note that the worst running time is only quadratic.

Related work on pattern analysis of RNAs focuses on RNAs with secondary structures. That is, if RNAs are represented as graphs preserving a linear vertex order, then the base-pair edges which are drawn in the upper halfplane do not cross. These graphs are often represented as trees implying numerous applicable algorithms in reasonable time and space. Wang et al. (1998) published an algorithm for finding a largest approximately common substructure between two trees. This is an inexact pattern matching algorithm suitable for RNA secondary structures. A survey of computing similarity between RNAs with and without secondary structures until 1995 is given in Bafna et al. (1995). Gramm et al. (2002) formulated the arc-preserving problem: given two nested RNAs S_1 and S_2 with lengths n and m $(n \geq m)$, respectively, does S_2 occur in S_1 such that S_2 can be obtained by deleting bases from S_1 with the property that the arcs are preserved? This problem cannot be seen as biologically motivated because the structure of S_2 would be found split in S_1. It has been shown in Jiang et al. (2000) that finding the longest common arc-preserving subsequence for arc-annotated sequences (LAPCS), where at least one of them has a crossing arc structure is MAXSNP-hard. Exact pattern matching on RNAs has been done by P. Gendron and Major (1998). They propose a backtracking algorithm similar to an algorithm from Ullmann (1976) solving the subgraph isomorphism problem from graph theory. They developed an algorithm aimed at finding recurrent patterns in one RNA. However, the running time allows the execution only on short RNAs. In analogy to RNAs, the comparison between two proteins with regard to patterns has been studied by Viksna and Gilbert (2001). They presented a pattern matching and pattern discovery algorithm of TOPS diagrams (description of protein topologies) which also has an exponential running time.

The paper is organized as follows: In Section 2, we introduce the reader into definitions and notations of RNAs. We define patterns and matchings between two RNAs such that a maximally extended pattern can be described. In Section 3, we present our fast dynamic programming method which operates on sequential as well as structural properties of RNAs. In Section 4, we illustrate our algorithm by comparing two Hepatitis C virus internal ribosome entry sites (IRES) sharing a lot of local sequential and structural patterns and by detecting a strongly conserved region in SECIS elements in *Methanococcus jannaschii*.

2. DEFINITIONS AND NOTATIONS

The primary sequence of a ribonucleic acid (RNA) is a sequence of nucleotides over the alphabet $\Sigma = \{A, C, G, U\}$. $S_i(j)$ denotes the nucleotide at position j in sequence i. A base is part of a nucleotide. The secondary or tertiary structure is given by a list of base-pairs leading to structure formation. A base-pair is given by a tuple (i_1, i_2) of positions meaning that the bases at positions i_1 and i_2 form hydrogen bonds. The following restriction holds for secondary structures: for any two base-pairs (i_1, i_2) and (j_1, j_2) either $i_1 < i_2 < j_1 < j_2$ or $i_1 < j_1 < j_2 < i_2$. A tuple (S, P) describes an RNA consisting of the primary sequence S and the base-pair list P.

A higher level of structure interpretation is given by a classification of structure elements; we call them loops. A loop is enclosed by a base-pair (i_1, i_2). It contains all nucleotides lying on the path starting at position $i_1 + 1$ and seeking the shortest way over phosphodiester and hydrogen bonds to position $i_2 - 1$. On the way, the bonds are taken only once, and only those nucleotides are chosen that lie in the inner part of the RNA,

i.e., at positions $i_1 + 1$ or higher and positions $i_2 - 1$ or less. We distinguish the following loops: a stack consists of one base-pair stacked on another one, a left/right bulge is a stacked base-pair containing single stranded nucleotides on the left/right side within these base-pairs, an internal loop contains a base-pair with single stranded nucleotides on both sides, a hairpin occurs at the end of stems and encloses single stranded nucleotides, and finally, there is a multi-branched loop from which three or more stems radiate (Fig. 1).

Definition 1 (Path). *A path in an RNA (S, P) from a nucleotide at position i to a nucleotide at position j is a sequence of positions $< p_1, p_2, ..., p_k >$ such that $p_1 = i$, $p_k = j$ and there exist phosphodiester or hydrogen bonds between $S(p_{l-1})$ and $S(p_l)$ for $l = 2, \ldots, k$.*

Definition 2 (Connected Pattern). *A connected pattern of size k in an RNA (S, P) is a set of positions $T = \{p_1, p_2, ...p_k\}$ such that for any two nucleotides $S(p_i)$ and $S(p_j)$, $p_i, p_j \in T$, there exists a path from $S(p_i)$ to $S(p_j)$, completely lying in T.*

Proposition 1. *Two connected patterns in an RNA (S, P) are given by $T_1 = \{p_{1_1}, p_{1_2}, ...p_{1_k}\}$ and $T_2 = \{p_{2_1}, p_{2_2}, ...p_{2_l}\}$. If there exists at least one position p_i such that $p_i \in T_1$ and $p_i \in T_2$, then the union of the two connected patterns is connected.*

Proof. Suppose there exists a position p_i with $p_i \in T_1$ and $p_i \in T_2$. For any nucleotide at position $p_r \in T_1$, there exists a path $< p_r, \ldots, p_i >$ such that each nucleotide on the path is lying in the first connected pattern. The same also holds for the second connected pattern. Hence, there exists a path from any nucleotide at position $p_r \in T_1$ to p_i to any nucleotide at position $p_s \in T_2$. ■

For the comparison of patterns between two RNAs we need the following definition:

Definition 3 (partial matching). *A partial matching of two RNAs $R_1 = (S_1, P_1)$ and $R_2 = (S_2, P_2)$ is given by a set M of pairs (i, j) such that for any i there exists exactly one j (and vice versa) with*

$$M = \{(i, j) | S_1(i) = S_2(j) \wedge STR_1(i) = STR_2(j)\}$$

The first condition in M ensures that the nucleotides are equal. The second condition consists of a function $STR_k(i)$ which returns the structure type of a nucleotide at position i in sequence k ($k = 1, 2$). We make use of three structure types of a nucleotide: single stranded (ss), i.e. this nucleotide is not involved in any base-pairing interaction, left paired (lp), i.e. the nucleotide is involved in a base-pairing and the base-pair partner is located at a higher position, and right paired (rp); i.e., the base-pair partner has a lower position. We are interested in finding partial matchings of maximally extended, connected patterns between two RNAs. A pattern is maximally extended if there exists no pattern which has this maximally extended pattern as a proper sub-pattern under a partial matching.

We call loops which are enclosed by base-pairs inner loops. An array of nucleotides in an inner loop is a sequence of nucleotides which are connected due to their loop positions. An example of loop positions is shown in Figure 3.

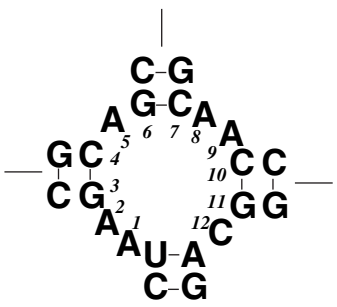

FIG . 3. Position numbering in a multi-branched loop. The positions from 1 to 12 indicate a loop walk.

3. DYNAMIC PROGRAMMING ALGORITHM

In this section, we study the problem of finding maximally extended, common patterns for each combination of positions i in the first RNA and j in the second RNA. We propose an efficient dynamic programming algorithm applied to RNAs with at most secondary structures. The positions serve as starting points from which a pattern is maximally extendable. The key idea is to maintain three $n \times m$ matrices M^{eb}, M^{nb}, and M^{loop}, where n and m are the lengths of the RNAs, respectively. The matrices correspond to calling subroutines which treat all cases of matchings: base-pair matching, almost one base matching involved in a base-pairing and matching of inner loops. The algorithm works from inside to outside according to the base-pair lists. We introduce an auxiliary function *max_matching* which returns the size of a maximally extended pattern found so far in inner loops.

$bp_k(i)$ denotes the position of the base-pair partner of $S_k(i)$. The positions of an inner loop which is enclosed by a base-pair $(i, bp(i))$ is given as $< l_1, l_2, \ldots, l_{size} >$. We call the positions a loop walk according to the order the nucleotides are considered. This holds for all kinds of loops: hairpins, stacks, left/right bulges, internal loops or multi-branched loops. An example of a loop walk in a multi-branched loop is given in Figure 3.

3.1. Auxiliary function

An important auxiliary function of our algorithm is *max_matching(i,j,r)*. This function is performed on inner loops starting at positions i in the first RNA and j in the second RNA with size r of a common array of nucleotides in inner loops. It returns the size of the new, common pattern found so far (Fig. 4).

The auxiliary function makes use of matrix entries in M^{eb} and M^{nb}. They are already pre-computed due to the inside to outside step. The function is called with a parameter r which is always less than the size to the end of the current inner loop clockwise. Line 2 checks whether the nucleotides are equal. If not, then the current size is returned. If yes, then the nucleotides are checked whether they have the same structure type (line 3,5,10). If both nucleotides are non-paired(ss), then the pattern size is increased by one. Line 5 checks whether the nucleotides are left-paired. Either the base-pair partners are equal, too, then we have a base-pair matching and the pre-computed value is in M^{eb} (line 7) or the base-pair partners are not equal, i.e. the pre-computed value is in M^{nb} (line 9). In the latter case, we know that the connectivity of a pattern is broken; the procedure stops extending this pattern by returning the current size. The situation given in line 10 can only occur if $\mathrm{r} = 0$; i.e., we know that the left base-pair partners are not equal since the call of this procedure is only done to the right base-pair partners from the procedure *loop_walking*. The positions in this function operate on loop positions. The *pos* function returns global positions of RNAs.

3.2. Loop walking

Loop walking computes the sizes of maximally extended, common patterns in inner loops. It is called from the main procedure with positions i and j in the first and second RNA, respectively (Fig. 5).

The values of $l_{i_{size}}$ and $l_{j_{size}}$ are the numbers of nucleotides in inner loops to the end of inner loops considering them clockwise. They can be easily obtained by traversing the inner loop in advance. The while-loop in line 6 determines the sizes of common nucleotide arrays in inner loops including structure type checking (line 8). The sizes of the new common patterns found so far will be given by the procedure *max_matching* (line 10,11). This step is only done if the tuples (k,l) are not yet considered. This means that if the size of the same nucleotide array starting at positions k and l is at least 2, then the value of $M^{loop}(pos(l+1), pos(k+1))$ need not be recomputed since it has already been considered. This information can be easily stored in a binary $n \times m$ matrix. Note again, that the procedure operates on loop positions.

3.3. Base-pair matching

Here, we assume that a base-pair $(i, bp(i))$ in the first RNA is matched with a base-pair $(j, bp(j))$ in the second RNA. We distinguish two cases of obtaining the correct size for the maximally extended pattern. The computation takes place at $M^{eb}(i, j)$. We omit the algorithmic code here.

MAX_MATCHING(i, j, r)

1 $size \leftarrow 0, m \leftarrow 0$

2 **while** $m \leq r$ *and* $S_1(i+m) = S_2(j+m)$

3 **do if** $STR_1(i+m) = ss$ *and* $STR_2(j+m) = ss$

4 **then** $size \leftarrow size + 1$

5 **else if** $STR_1(i+m) = lp$ *and* $STR_2(j+m) = lp$

6 **then if** $S_1(i+m+1) = S_2(j+m+1)$

7 **then** $size \leftarrow size + M^{eb}(pos(i+m), pos(j+m))$

8 $m \leftarrow m+1$

9 **else** *return* $size + M^{nb}(pos(i+m), pos(j+m))$

10 **else if** $STR_1(i+m) = rp$ *and* $STR_2(j+m) = rp$

11 **then** $size \leftarrow size + M^{nb}(pos(i+m), pos(j+m))$

12 $m \leftarrow m+1$

13 **return** $size$

FIG . 4. Auxiliary function for computing the size of a maximally extended pattern in inner loops between two RNAs.

Case 1: The maximal, common array of nucleotides in inner loops to the right of positions i and j and to the left of bp(i) and bp(j) do not overlap (Fig. 6).

The value of $M^{eb}(i, j)$ is given as $M^{loop}(i+1, j+1) + M^{loop}(i', j') + 2$, where i' and j' are the first positions left, i.e. counter-clockwise, to $bp(i)$ and $bp(j)$ sharing the same common array of nucleotides from i' to $bp(i) - 1$ and from j' to $bp(j) - 1$. Either the size of the array is greater than 1, then $M^{loop}(i', j') \neq 0$, otherwise we set $M^{loop}(i', j') = 0$ in the above sum. Finally, the matrix entries of $M^{loop}(i+1, j+1)$ and $M^{loop}(i', j')$ are set to 0 since the sizes of these patterns are now included in $M^{eb}(i, j)$. This step is necessary to prevent the output of proper subpattern. The base-pair matching, i.e. (i, j) matches (i', j'), is counted as a pattern of size 2.

Case 2: The maximal, common array of nucleotides in inner loops to the right of positions i and j and to the left of bp(i) and bp(j) do overlap in at least one RNA (Fig. 7).

As shown in Figure 7, there is no clear decision how the nucleotides should be matched. In the left figure, the two A's immediately before the cutting line are able to match the rightmost A's or the leftmost A's or one leftmost and one rightmost A in the right figure. This is the only situation where the entire assignment becomes ambiguous, and hence, we get an exponential number of solutions if all matchings were considered. It would result in a non-tractable pattern problem. Fortunately, a maximum pattern can be extracted by looking first at which sub-array of nucleotides in the cycle is able to match to both sides of the other inner loop. In our example, this sub-array consists of the two A's. It can be obtained by traversing the inner loops from positions $i+1$ and $j+1$ and by matching their corresponding nucleotides until no matching is possible any more or

LOOP_WALKING(i, j)

1 $l_{i_1} \leftarrow 1 (= global\, position\, i)$

2 $l_{j_1} \leftarrow 1 (= global\, position\, j)$

3 **for** $k \leftarrow l_{i_1}$ **to** $l_{i_{size}}$

4 **do for** $l \leftarrow l_{j_1}$ **to** $l_{j_{size}}$

5 **do** $r \leftarrow 0$

6 **if** (k, l) *not yet considered*

7 **then while** $k + r < l_{i_{size}} \wedge\ l + r < l_{j_{size}}$

8 $\wedge\quad S_1(k + r) = S_2(l + r)$

9 $\wedge\quad STR_1(k + r) = STR_2(l + r)$

10 **do** $r \leftarrow r + 1$

11 $M^{loop}(pos(l), pos(k)) \leftarrow$

12 $max\_matching(k, l, r)$

FIG . 5. *Loop walking* computes the size of a maximally extended, common patterns in an inner loop starting at position i in the first RNA and j in the second RNA.

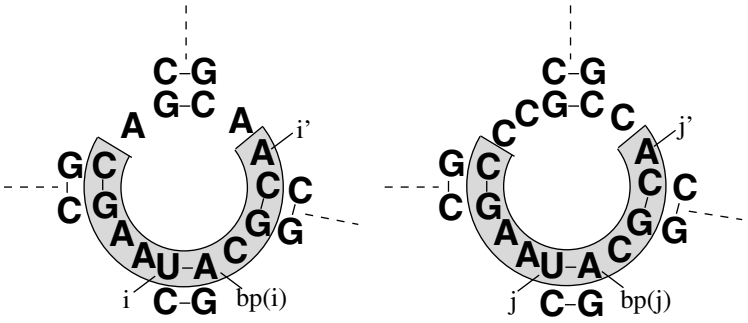

FIG . 6. The base-pairs $(i, bp(i))$ and $(j, bp(j))$ match. The first case is given by the non-overlapping common array of nucleotides in inner loops to the right of positions i and j and to the left of bp(i) and bp(j).

the positions $bp(i) - 1$ or $bp(j) - 1$ are reached. The same procedure is performed on positions $bp(i) - 1$ and $bp(j) - 1$ counter-clockwise. The overlapping array of nucleotides is ambiguous to match. We proceed by going from one nucleotide to the next in the sub-array and make a cut such that the right/left side of the first RNA is matched to the right/left side of the second RNA. For both sides, the *max_matching* values are

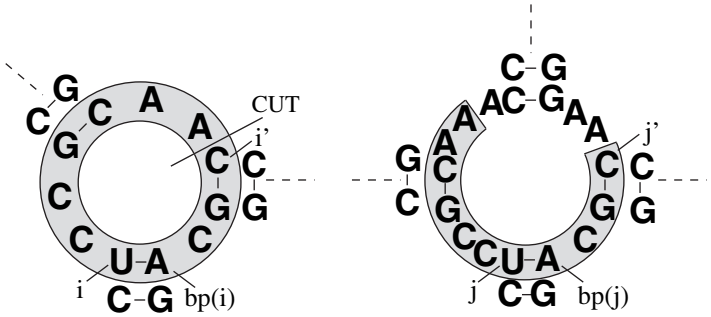

FIG . 7. The base-pairs $(i, bp(i))$ and $(j, bp(j))$ match. The second case is given by the overlapping common array of nucleotides in inner loops to the right of positions i and j and to the left of bp(i) and bp(j).

computed. Since we don't want a cubic time algorithm, we add and subtract the corresponding *max_matching* values of the current nucleotides in the sub-array. This prevents having to re-compute the whole *max_matching* values. Note that base-pairs are not broken; i.e., if there is a base-pair matching in inner loops, then the step is to jump over the base-pair partner. The cut which provides a maximum value is then chosen; i' and j' are the positions immediately after the cut. The value of $M^{eb}(i, j)$ is given as $M^{loop}(i+1, j+1)+M^{loop}(i', j')+2$. As in case 1, the matrix entries of $M^{loop}(i + 1, j + 1)$ and $M^{loop}(i', j')$ are set to 0. Of course, both inner loops may have the overlapping situation, but it does not affect the time complexity.

3.4. None base-pair matching

The last matching case is when at most one base of a base-pairing in one RNA is matched with at most one base of the same structure type in the other RNA. Thus, overlapping regions can be excluded. Entries of M^{nb} can be computed uniquely. The algorithmic code is given in Figure 8.

Line 1 checks if the bases of structure type lp are the same. If it succeeds, then the entry of $M^{nb}(i, j)$ is computed. The same holds for the bases with structure type rp (line 4). The procedure is called, if the nucleotide at position i is involved in a left base-pairing in the first RNA and the nucleotide at position j in a left base-pairing in the second RNA. i' and j' are the leftmost positions such that $M^{loop}(i', j') \neq 0$, if both RNAs share the same common array of nucleotides in inner loops from i' to $bp(i)$ and from j' to $bp(j)$; otherwise, $M^{nb}(bp(i), bp(j))$ is set to 1.

NONE_BASE-PAIR_MATCH(i, j)

1 **if** $S_1(i) = S_2(j)$

2 **then** $M^{nb}(i, j) = M^{loop}(i + 1, j + 1) + 1$

3 $M^{loop}(i + 1, j + 1) = 0$

4 **else if** $S_1(bp(i)) = S_2(bp(j))$

5 **then** $M^{nb}(bp(i), bp(j)) = M^{loop}(i', j') + 1$

6 $M^{loop}(i', j') = 0$

FIG . 8. Algorithm for the none base-pair matching case.

3.5. Main procedure

The main procedure performs the pattern search from inner to outer loops. It consists of taking base-pairs $(i, bp(i))$ from the first and $(j, bp(j))$ from the second RNA. A call on *loop_walking*$(i+1, j+1)$ (Section 3.2) is executed. Depending on the base-pair matching cases, the main procedure proceeds by calling either the *base-pair_match(i,j)* procedure (Section 3.3) or the *none-base-pair(i,j)* matching procedure (Section 3.4). A special kind of an inner loop is the external loop starting at the first position and ending at the last position. Here, we assume that both RNAs are enclosed by a virtual base-pair in order to execute the *loop_walking* procedure, but the remaining procedures are not executed any more.

3.6. Traceback

We have stored all sizes of maximally extended, common patterns in M^{loop}. The starting positions i and j are given by entries $M^{loop}(i, j) \neq 0$. The traceback retrieves maximally extended patterns using the same cases of matching base-pairs, matching none base-pairs and matching arrays of nucleotides in inner loops. One can make use of an additional $n \times m$ matrix, which stores the cuts at the appropriate positions during computation. Then the traceback step for one pattern will not be worse than linear time.

3.7. Complexity

Time complexity. The main procedure consists of running through all combinations of base-pairs in both RNAs. This yields a running time of $O(nm)$. For each combination the *loop_walking* procedure is executed only once. The procedure operates on each combination of nucleotide positions in inner loops. Since inner loops do not overlap each other in one RNA, this does not increase the running time. The *loop_walking* procedure calls the *max_matching* procedure which also operates on inner loop positions. The *max_matching* procedure marks nucleotides as considered if they are matched successfully. This is noticed by the *loop_walking* procedure such that both procedures consider each combination of inner loop positions almost twice. Therefore, the running time of $O(nm)$ is retained. The *base-pair_match* procedure consists of finding two common arrays of nucleotides in both directions of a base-pair combination. Finding the cutting line means to traverse the overlapping regions. This does not cost more than the number of nucleotides in inner loops. The *none-base-pair* procedure has cost of finding a common array of nucleotides from the right base-pair partner. Therefore, we yield a running time of $O(nm)$.

Space complexity. The algorithm needs three $n \times m$ matrices M^{eb}, M^{nb} and M^{loop} for computing the sizes, one $n \times m$ boolean matrix for marking considered nucleotides (Section 3.2) and one $n \times m$ matrix for storing positions of cuts. The space complexity is thus given by $O(nm)$.

4. EXPERIMENTAL RESULTS

The algorithm is implemented in C++ and can be obtained by contacting one of the authors. The output is a sorted list of patterns according to their descending sizes. The sorting algorithm dominates the running time, but it can be omitted if desired.

We have performed two tests showing that (i) although global structures of RNAs diverge, they share a lot of local sequence structure properties and (ii) strong conserved regions of several RNAs can be easily detected by comparing them pairwise.

The first test is performed on Hepatitis C virus internal ribosome entry sites (IRES). The RNAs are taken from the RNaseP Database Brown (1999) given by their accession codes: (i) AF165050 and (ii) D45172. Their sequence lengths are 391 and 379, respectively. We have folded both RNA sequences into their optimal structures with the use of RNAfold from the Vienna RNA Package 1.4 Hofacker et al. (1994). The two RNAs are shown in Figure 9. At first sight, their global structures differ enormously. Our common pattern detection algorithm exhibits big similarities in both RNAs. We have highlighted the five largest common patterns. We can see that both RNAs share a lot of sequential and structural patterns although we have taken

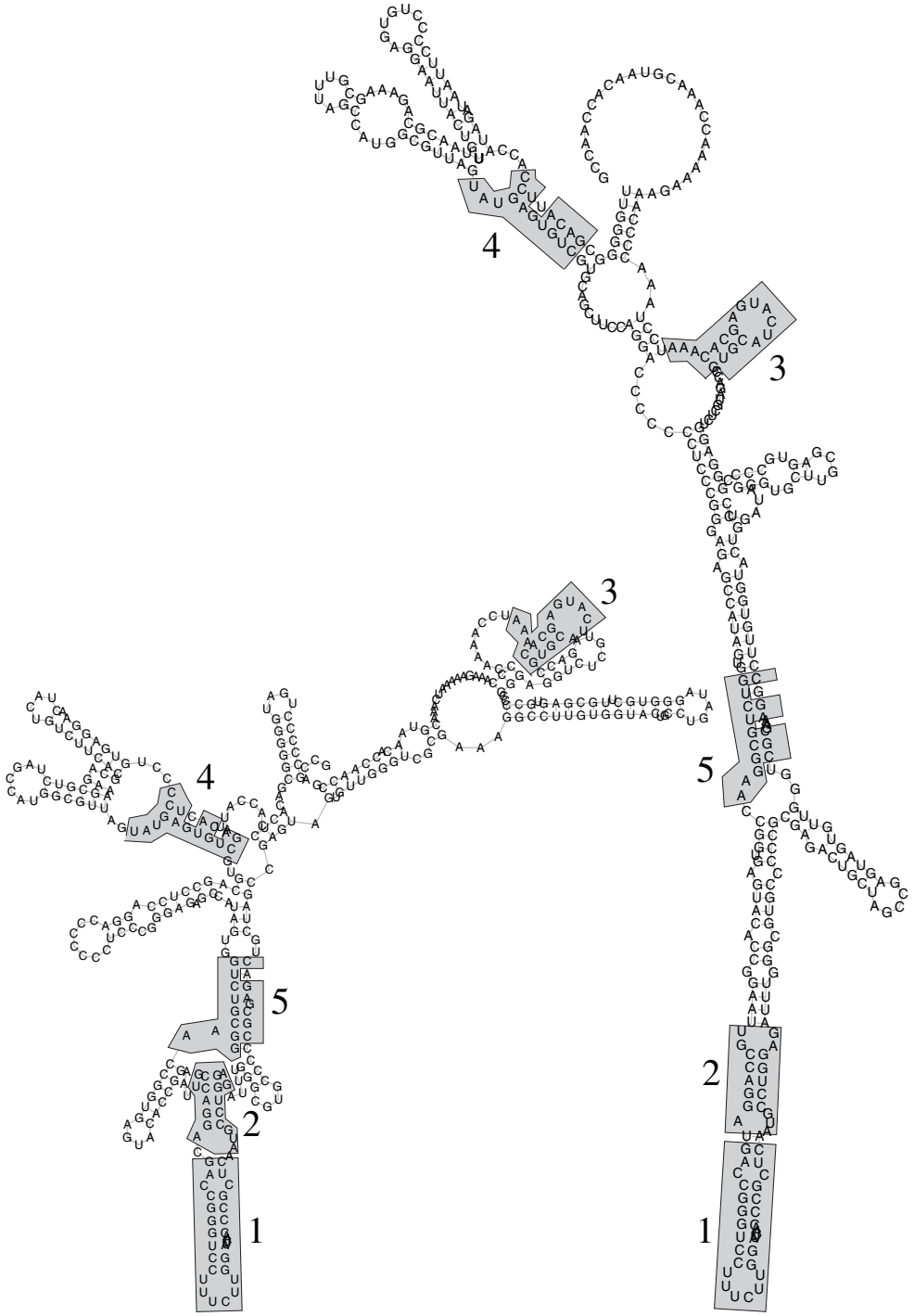

FIG . 9. Hepatitis C virus internal ribosome entry sites (IRES) of two RNAs. They are given by their accession codes: D45172 (left RNA) and AF165050 (right RNA). The five largest patterns in terms of their sequential and structural properties are highlighted. The ascending numbering of patterns is equivalent to the descending size of patterns. The pattern numbered as 1 is the maximum common pattern.

only the structures with minimum free energy. But these local patterns occur in suboptimal structures as well (data not shown). The size of the maximum common pattern is 30. Altogether, the algorithm has found 8829 patterns.

The second test was performed on putative SECIS-elements (Selenocysteine Insertion Sequence) in non-coding regions of *Methanococcus jannaschii*. They were taken from Wilting et al. (1997). By pairwise comparison we yielded a strongly conserved region which coincides with the results found manually by Wilting et al. (1997). These patterns are the maximally sized patterns (Fig. 10).

5. CONCLUSION

We have presented an efficient dynamic programming approach of detecting common sequence structure patterns between two RNAs in time $O(nm)$ and space $O(nm)$. The output are maximally extended, bond-preserving patterns. We omit the output of proper sub-patterns since this increases the running time exponentially and they are not suited for detecting the most similar regions in RNAs. The bonds of base-pairs are preserved due to the fact, that if they were not preserved then we would allow matchings of the same bases involved in different base-pairings. Have a look at the example shown in Figure 2 again. There, we get a matching case other than depicted, namely that the U above the shaded box in the right RNA matches the U in the left dashed box in the left RNA since both U's are left paired, and the base-pair partner A in the right RNA matches the A in the right dashed box of the left RNA since both bases are right paired. This situation is prohibited because these kinds of base-pair breakings are very unlikely to occur and, thus, it makes no sense biologically. Furthermore, it would increase the running time of our algorithm. The bond preserving restriction also avoids finding mirrored patterns; i.e., a helix of CG pairs does not match a helix of GC pairs.

Beside the detection of exact patterns, there are various possibilities to extend and to retrieve new results for particular applications. One of them might be to align sequences globally based on local sequential and structural patterns. Here, the crucial point is to choose those patterns such that the sequential substrings resulting from these patterns can be arranged without overlappings. One can think of using as many local patterns as possible. This idea can also be applied to local regions of RNAs. First, an accumulation of local RNA patterns has to be found, and then using the patterns for locally aligning these overall regions with the help of a local alignment approach as proposed, for example, in Backofen and Will (2004) with time complexity $O(n^2m^3)$, where n and m are the lengths of the detected regions. This idea exhibits strong homologous regions based on sequence and structure patterns. The main advantage is, of course, that large RNAs (i.e., RNAs with several thousand bases) can be investigated in reasonable time. Therefore, future work will be investigating approaches for global and local alignments based on sequential and structural features. In addition to local alignments, it is easier to describe local regions as motifs with secondary structures. They will be helpful in recognizing, for example, functional regions.

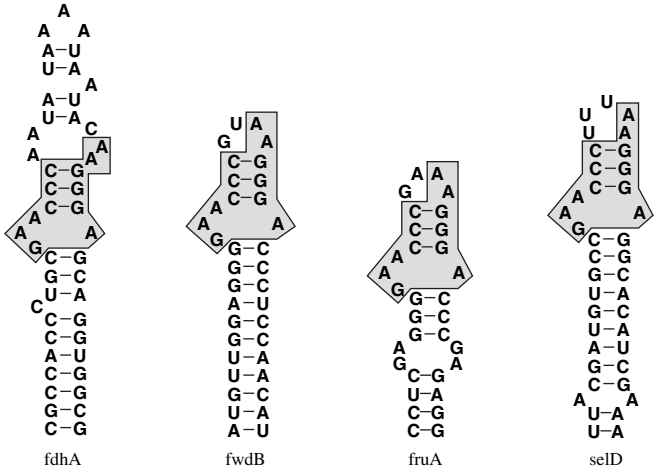

FIG . 10. Putative SECIS-elements in non-coding regions of *Methanococcus jannaschii*. The algorithm detects a strongly conserved pattern by pairwise comparisons. This pattern was also discovered by hand from (Wilting et al., 1997).

ACKNOWLEDGMENT

This work was supported by the DFG within the national project "Selenoproteine."

REFERENCES

Backofen, R. and Will, S.: 2004, *Journal of Bioinformatics and Computational Biology (JBCB)* 2(4), 681

Bafna, V., Muthukrishnan, S., and Ravi, R.: 1995, in *Proc. 6th Symp. Combinatorical Pattern Matching*

Brown, J. W.: 1999, *NAR* 27(1)

Eppstein, D.: 1999, *J. Graph Algorithms & Applications* 3(3), 1

Gilbert, D., Westhead, D., and Viksna, J.: 2003, *Artificial Intelligence and Heuristic Methods in Bioinformatics* pp 128–147

Gilbert, D., Westhead, D., Viksna, J., and Thornton, J.: 2001, *Computational Chemistry* 26(1), 23

Gramm, Guo, and Niedermeier: 2002, *Foundations of Software Technology and Theoretical Computer Science* 22

Gusfield, D.: 1997, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.*, CUP

Hofacker, I., Fekete, M., Flamm, C., Huynen, M., Rauscher, S., Stolorz, P., and Stadler, P.: 1998, *Nucleic Acids Res* 26(16), 3825

Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, S., Tacker, M., and Schuster, P.: 1994, *Monatshefte Chemie* 125, 167

Jiang, T., Lin, G., Ma, B., and Zhang, K.: 2002, *Journal of Computational Biology* 9(2), 371

Jiang, T., Lin, G.-H., Ma, B., and Zhang, K.: 2000, in *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching (CPM2000)*

Matula, D. W.: 1968, *SIAM Rev.* 10, 273

Messmer and Bunke: 2000, *IEEE Transactions on Knowledge and Data Engineering* 12

Messmer, B. T. and Bunke, H.: 1997, *Technical Report*

P. Gendron, D. G. and Major, F.: 1998, *High Performance Computing Systems and Applications*

Pavesi, G., Mauri, G., and Pesole, G.: 2004, *Journal of Computer Science and Technology* 19, Issue 1, 2

Shamir and Tsur: 1997, in *ISTCS: 5th Israeli Symposium on the Theory of Computing and Systems*

Ullmann, J. R.: 1976, *Journal of the ACM* 23(1), 31

Viksna, J. and Gilbert, D.: 2001, in *Proceedings of the First International Workshop on Algorithms in Bioinformatics (WABI 2001)*, No. 2149 in LNCS, pp 98–111

Wang, J. T.-L., Shapiro, B. A., Shasha, D., Zhang, K., and Currey, K. M.: 1998, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8), 889

Wilting, R., Schorling, S., Persson, B. C., and Boeck, A.: 1997, *Journal of Molecular Biology* 266(4), 637

Address reprint requests to:
Dr. Sven Sibert
Department of Bioinformatics
Institute of Computer Science
Albert-Ludwigs-University Freiburg
Georges-Koehler-Allee 106
79110 Freiburg, Germany

E-mail: siebert@informatik.uni-freiburg.de