Title	An Efficient Alignment Algorithm for Searching Simple Pseudoknots over Long Genomic Sequence
Author(s)	Ma, CCC; Wong, KF; Lam, TW; Hon, W; Sadakane, K; Yiu, SM
Citation	IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2012, PP, Issue: 99, p. 1
Issued Date	2012
URL	http://hdl.handle.net/10722/165841
Rights	Creative Commons: Attribution 3.0 Hong Kong License

An Efficient Alignment Algorithm for Searching Simple Pseudoknots over Long Genomic Sequence

Christopher Ma, Thomas K.F. Wong, T.W. Lam, W.K. Hon, K. Sadakane, and S.M. Yiu

Abstract—Structural alignment has been shown to be an effective computational method to identify structural noncoding RNA (ncRNA) candidates as ncRNAs are known to be conserved in secondary structures. However, the complexity of the structural alignment algorithms becomes higher when the structure has pseudoknots. Even for the simplest type of pseudoknots (simple pseudoknots), the fastest algorithm runs in $O(mn^3)$ time, where m, n are the length of the query ncRNA (with known structure) and the length of the target sequence (with unknown structure), respectively. In practice, we are usually given a long DNA sequence and we try to locate regions in the sequence for possible candidates of a particular ncRNA. Thus, we need to run the structural alignment algorithm on every possible region in the long sequence. For example, finding candidates for a known ncRNA of length 100 on a sequence of length 50,000, it takes more than one day. In this paper, we provide an efficient algorithm to solve the problem for simple pseudoknots and it is shown to be 10 times faster. The speedup stems from an effective pruning strategy consisting of the computation of a lower bound score for the optimal alignment and an estimation of the maximum score that a candidate can achieve to decide whether to prune the current candidate or not.

Index Terms—Noncoding RNAs, pseudoknot, structural alignment

1 Introduction

A NON-CODING RNA (ncRNA) is a functional RNA molecule which is not translated into a protein. There are many different types of ncRNAs such as tRNAs, rRNAs, snoRNAs, microRNAs, and siRNAs. These RNA molecules have been found to be involved in many biological processes. The number of ncRNAs within the genome was underestimated before, but recently some databases reveal over 1,973 ncRNA families [1]. Data accumulated on ncRNAs and these families show that ncRNAs may be as diverse as protein molecules.

Identifying ncRNAs is an important problem. It is known that the structure of an ncRNA molecule usually plays an important role in its biological functions. Some researchers attempted to identify ncRNAs by considering the stability of secondary structures formed by substrings of a given genome [2]. Although, on average, the functional RNAs have lower folding energy than the random sequences of the same length and dinucleotide frequency [3], this method is not quite effective because a random sequence with high GC composition also allows an energetically favorable secondary structure [4].

- *C. Ma, T.K.F. Wong, T.W. Lam, and S.M. Yiu are with the Department of Computer Science, The University of Hong Kong, Rm 301, Chow Yei Ching Building, Pokfulam Road, Hong Kong. E-mail: {cccma, kfwong, twlam, smyiu}@cs.hku.hk.*
- *W.K. Hon is with the Department of Computer Science, National Tsing Hua University, 101 Kuang Fu Road, Section 2, Hsinchu, Taiwan 300, R.O.C. E-mail: wkhon@cs.nthu.edu.tw.*
- *K. Sadakane is with the National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan. E-mail: sada@nii.ac.jp.*

Manuscript received 9 Sept. 2011; revised 30 Apr. 2011; accepted 1 July 2012; published online 23 July 2012.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2011-09-0230. Digital Object Identifier no. 10.1109/TCBB.2012.104.

Another promising method is the comparative approach. The idea is to make use of a known ncRNA and try to identify ncRNA candidates along the genome whose sequence and structure are similar to that of the ncRNA. The resulting regions are the potential ncRNAs candidates of the same family. The key of this approach is to compute the structural alignment between the folded ncRNA (query) and the unfolded region (target). The unfolded sequence will be aligned to the folded ncRNA. The alignment score represents their sequence and structural similarity. RSEARCH [5], FASTR [6], and INFERNAL [7] for Rfam were developed based on this approach.

However, these tools [5], [6], [7] do not support pseudoknots. Given two base-pairs at positions (i, j) and (i', j'), where $i < j$ and $i' < j'$, pseudoknots are base-pairs crossing each other, i.e., $i < i' < j < j'$ or $i' < i < j' < j$. In some studies, secondary structures including pseudoknots are found involved in some functions such as telomerase [8], catalytic functions [9], and selfsplicing introns [10]. For example, the pseudoknot structure in the human telomerase RNA is conserved in all vertebrates and is essential for telomerase activity [11]. Structural alignment of ncRNAs containing pseudoknots is believed to be NP-complete and the problem is computationally harder.

Matsui et al. developed a method of computing the structural alignment to support some complicated pseudoknot structures which they refer as *2-crossing* [12] . Their algorithm runs in $O(mn^5)$ with space complexity of $O(mn^4)$ where m is the length of the query sequence and n is the length of the target sequence. However, their method can only consider the structural similarity but not sequence similarity between the query and the target sequence.

TABLE 1
Comparison on the Running Time between Our Method and Han's

Family ID	Family name	Query length	Number of members embedded	Our running time (seconds)	Han's running time (seconds)	Speedup
RF00165	Corona_pk3	62	30	2220	16213	7.3
RF00390	UPSK	23	25	1215	4613	3.8
RF00505	RydC	66	75	2058	17489	8.5
RF00507	Corona_FSE	85	23	4064	37350	9.2
RF01072	TMV_UPD-PK3	21	40	1124	4471	4.0
RF01074	RF_site1	40	12	1731	10025	5.8
RF01076	RF_site2	72	38	2670	23990	9.0
RF01080	UPD-Pkib	32	16	1543	7845	5.1
RF01081	SBWMV1_UPD-Pke	26	14	1341	7418	5.5
RF01087	PK-repZ	148	37	25725	284186	11.0
RF01089	PK-repBA	116	25	10481	114237	10.9
RF01093	RF_site5	60	38	2274	15899	7.0
RF01095	PK-CuYV_BPYV	56	4	2132	15127	7.1
RF01097	RF_site8	52	15	2188	14635	6.7
RF01100	PK-BYV	40	4	1697	9997	5.9
RF01102	PK1-TEV_CVMV	36	8	1524	9477	6.2
RF01104	SBWMV2_UPD-PKb	29	8	1488	7334	4.9
RF01109	SBRMV1_UPD-PKd	22	12	1185	4689	4.0
RF01118	PK-G12rRNA	124	44	17714	184441	10.4

In this paper, we consider the structural alignment problem as follows: given a query sequence with pseudoknot structure and a target sequence with unknown structure, our aim is to output the alignment with maximum score between the whole query sequence and the whole target sequence. The score, which will be formally defined in Section 2.1, is computed according to both of their sequence and structural similarity. Higher score represents higher similarity between the two sequences according to their sequences and structures.

PAL, which was developed by Han et al. [13], can solve this problem in $O(mn^3)$ time with space complexity of $O(mn^3)$ for simple pseudoknot structure. For the definition of these pseudoknot structures, please refer to Section 2. However, their algorithm is still not feasible for long RNA sequences (say the length is over 50k) due to the extensive time required. For example, as shown in Table 1 for the pseudoknotted ncRNA family of sequence length around 100 to 150, when searching a long genome of length 50,000 to find regions of which the sequences and the structures are similar to that of the selected query of the family, it takes 1.3 to 3.3 days to finish. The figure shows that the tool seems impractical for searching ncRNAs along a long genome.

We proposed an efficient algorithm for solving the same structural alignment problem for simple pseudoknot. While maintaining the same sensitivity and specificity, the speed up achieved by our method ranges from 8.5 to 11 as compared to PAL when the query is longer than 65. For an ncRNA family with simple pseudoknot structure of length around 100 to 150, our method requires only a few hours for searching along the same long genome. The experimental result shows that our algorithm provides a more feasible solution for searching pseudoknotted ncRNAs along a long genome. Although the method presented in this paper is designed for simple pseudoknot, a similar technique can also be applied to other structural alignment algorithms that handle other pseudoknot types.

Please note that there exists another approach like [14] to compute the alignment between the query structure including pseudoknots and the target sequence. Their method requires $O(k^t N^2)$ time where N is the length of the target sequence, t is the width of the tree according to the query structure and a parameter k (for details, please refer to [14]). Their approach is a heuristic approach. The choice of k would affect the accuracy of the result and their method do not consider the loop regions (i.e., the regions which do not have any base pair) of the query sequence when performing the alignment. If loop regions are large, the accuracy of the result would be decreased.

2 Preliminaries

Let $A = a_1 a_2 \dots a_m$ be an RNA sequence and M be the secondary structure of A. M is represented as a set of base pairs (a_i, a_j), $1 \le i < j \le m$. Let $M_{x,y} \subseteq M$ be the set of base pairs in the subsequence $a_x a_{x+1} \dots a_y, 1 \le x < y \le m$. $M_{x,y} = \{(a_i, a_j) \in M | x \le i < j \le y\}$.

$M_{x,y}$ is a *regular structure*,[1] if there does not exist two pairs $(i,j),(k,l) \in M_{x,y}$ such that $i < k < j < l$ or $k < i < l < j$. Note that an empty set is considered as a regular structure (see Fig. 1a for an example).

$M_{x,y}$ is a *simple pseudoknot* if $\exists x < x_1, x_2 < y$ (x_1, x_2 are referred as *pivot points*) such that (see Fig. 1b for an example):

1. each $(i,j) \in M_{x,y}$ satisfies either $x \le i < x_1 \le j < x_2$ or $x_1 \le i < x_2 \le j \le y$; and
2. M_L and M_R are both regular where $M_L = \{(i,j) \in M_{x,y} | x \le i < x_1 \le j < x_2\}$ and $M_R = \{(i,j) \in M_{x,y} | x_1 \le i < x_2 \le j \le y\}$.

1. In the literature, regular structure is also called *nested structure*.

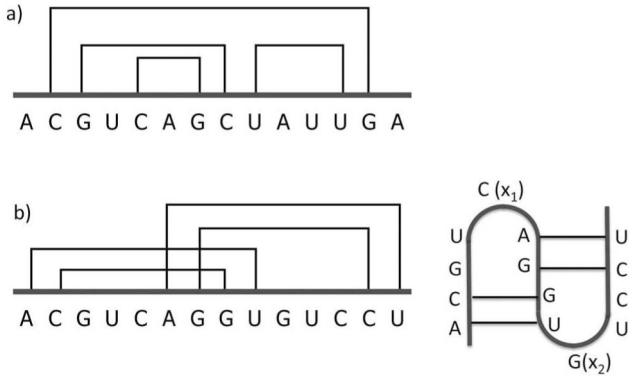

Fig. 1. (a) Regular structure. (b) Simple pseudoknot structure.

2.1 Structural Alignment

Let $S[1 \dots m]$ be a query sequence with known secondary structure M, and $T[1 \dots n]$ be a target sequence with unknown secondary structure. S and T are sequences from the character set $\{A, C, G, U\}$. A structural alignment between S and T can be represented by a pair of sequences $S'[1 \dots r]$ and $T'[1 \dots r]$ where $r \geq m, n$. S' is from S and T' is from T with spaces inserted in between the characters to make both sequences of the same length. A space cannot appear in the same position of S' and T'. The score of the alignment, which determines the sequence and structure similarity between S' and T', is defined as follows:

$$score = \sum_{i=1}^{r} \gamma(S'[i], T'[i]) + \sum_{\substack{i,j \text{ s.t.} \eta(i),\eta(j) \in M, \\ S'[i],S'[j],T'[i],T'[j] \neq '\_'}} \delta(S'[i], S'[j], T'[i], T'[j])$$

where $\eta(i)$ is the corresponding position in S according to the position i in S; $\gamma(t_1, t_2)$ and $\delta(x_1, y_1, x_2, y_2)$, where $t_1, t_2 \in \{A, C, G, U, '\_'\}$ and $x_1, x_2, y_1, y_2 \in \{A, C, G, U\}$, are the score for sequence similarity and the score for structural similarity, respectively. The calculation of structural alignment score is not restricted to any kind of secondary structure. It works in the same way for pseudoknot structure. The objective is to find an alignment such that the corresponding score is maximized.

Definition 1. *Optimal structural alignment between $S[1 \dots m]$ and $T[1 \dots n]$ is to find $S'[1 \dots r]$ and $T'[r \dots r]$, where $r \geq m, n$, S' is from S and T' is from T with spaces inserted in between the characters to make both sequences of the same length, such that the score of the structural alignment is maximum.*

Higher score represents higher similarity between the two sequences according to their sequences and structures. Also, if the score is high, then the alignment can reasonably reveal the secondary structure of the target sequence.

2.2 Sliding Window Technique and Semiglobal Alignment

In practice, when one would like to search regions whose sequences and structures are similar to those of the query sequence in a long genome ($T[1 \dots n]$), one would try to align every region on the genome, which is of similar length as the query's, and obtain those regions which result in high structural alignment scores between the query. Precisely,

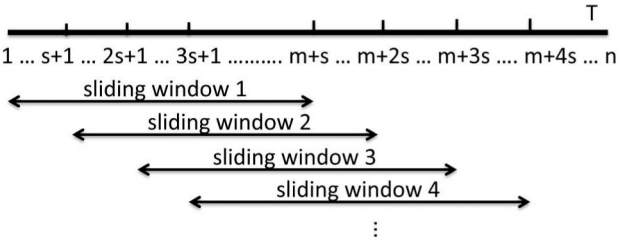

Fig. 2. The sliding windows on the sequence T: sliding distance is s and window size is $m + s$, assuming the maximum length of a candidate is m.

researchers usually use the sliding-window technique to perform the searching. If it is assumed that the maximum length of a candidate is m and there does not exist two candidates inside a region of length $m + s$, by setting the window size as $m + s$ and a sliding distance as s (see Fig. 2), for each subregion $T[1 \dots m + s]$, $T[s + 1 \dots m + 2s], \dots, T[ks + 1 \dots m + (k + 1)s] \dots$, where $1 \leq k \leq \lfloor (n - m)/s \rfloor - 1$, one could compute the *optimal semiglobal structural alignment* (which will be formally defined later) between the query and each of the subregions. This method is commonly used because one would only need to compute the structural alignment for every s position, and so there would lead to a speedup. However, according to our experiment, we found that even using this approach, it still takes a long time for a long genome sequence of length, say 50k.

The optimal semiglobal structural alignment problem is defined as follows.

Definition 2. *The optimal semiglobal structural alignment problem between $S[1 \dots m]$ and $T[1 \dots n]$ is to find a substring $T[p \dots q]$ where $1 \leq p, q \leq n$ (i.e., $T[p \dots q]$ is an empty string when $p > q$) such that the score of the optimal structural alignment between $S[1 \dots m]$ and the substring $T[p \dots q]$ is maximum.*

In the following sections, for the sake of simplicity, we regard each subregion $T[ks + 1 \dots m + (k + 1)s]$ as T during the illustration of the algorithms.

3 Algorithms

The optimal semiglobal structural alignment can be computed by using dynamic programming. The algorithm was developed by Han et al. [13] (although their method is designed to compute the optimal structural alignment, a minor modification on their method will be able to compute the optimal semiglobal structural alignment). The key idea behind their algorithm is the concept of a substructure which allows the optimal alignment to be computed recursively. The algorithm takes $O(mn^3)$ time where m is the length of the query sequence with known simple pseudoknot structure and n is the length of the target sequence with unknown structure.

3.1 Han's Algorithm

Let $S[i_0 \dots k_0]$ be the query sequence with simple pseudoknot structure M_{i_0,k_0}. As defined in Section 2, the pivot points x_1, x_2 for $S[i_0 \dots k_0]$ are known. Let $v = (i, j, k)$ be a triple with $i_0 \leq i < x_1 \leq j < x_2 \leq k \leq k_0$. Define the subregion $R(S, v) = [i_0 \dots i] \cup [j \dots k]$ (as shown in Fig. 3). Let $Struct(R) = \{(i, j) \in M | i, j \in R\}$ where R is a subregion. The subregion R defines a valid substructure ($Struct(R)$) of

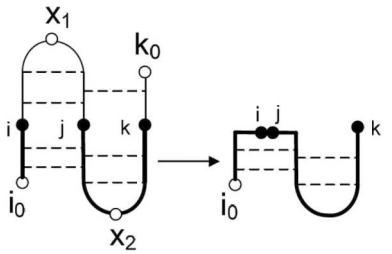

Fig. 3. Subregion $R(S,(i,j,k))$.

M if these does not exist $(i,j) \in M$ such that one endpoint of (i,j) is in R and the other is outside the region. Note that $Struct(R)$ is also a simple pseudoknot. Let $T[e_0 \dots g_0]$ be the target sequence with unknown structure. The definitions of R can also be applied to T. For any $v' = (e,f,g)$ where $e_0 \le e < f < g \le g_0$, the subregion $R(T,v') = [e_0 \dots e] \cup [f \dots g]$. Let $B(R_x,R_y)$ be the score of the optimal semiglobal alignment between a subregion R_x in S with substructure $Struct(R_x)$ and a subregion R_y in T. The score of the optimal semiglobal alignment between $S[i_0 \dots k_0]$ and $T[e_0 \dots g_0]$ can be obtained by setting $v^* = (x_1 - 1, x_1, k_0)$. The entry $\max_{e_0 \le e \le g_0} B(R(S,(x_1-1,x_1,k_0)), R(T(e,e+1, g_0)))$ provides the answer.

The value of $B(R_x,R_y)$ can be computed recursively [13]. Let $R_x = R(S,(i,j,k))$ and $R_y = R(T,(e,f,g))$. If (i,j) is a base pair in $Struct(R)$, there are four cases to consider. Case 1: MATCH_{both}—aligning the base pair (i,j) of S with (e,f) of T; Case 2: MATCH_{single}—aligning only one of the bases in (i,j) with the corresponding base in (e,f); Case 3: DELETE—deleting the base-pair (i,j) from S; Case 4: INSERT—inserting a space on S; Lemma 1 summarizes these cases [13]. Another situation, where (j,k) is a base pair, is similar.

Lemma 1. *Let $v = (i,j,k)$ and $v' = (e,f,g)$. $R_x = R(S,v)$ and $R_y = R(T,v')$. If (i,j) is a base pair, then according to [13], $B(R_x,R_y) =$ max*

$$
\begin{cases}
//\mathrm{MATCH}_{\mathrm{both}} \\
B(R(S,(i-1,j+1,k)),R(T,(e-1,f+1,g))) \\
\quad + \gamma(S[i],T[e]) + \gamma(S[j],T[f]) + \delta(S[i],S[j],T[e],T[f]); \\
//\mathrm{MATCH}_{\mathrm{single}} \\
B(R(S,(i-1,j+1,k)),R(T,(e-1,f,g))) + \gamma(S[i],T[e]) \\
\quad + \gamma(S[j],'\_'); \\
B(R(S,(i-1,j+1,k)),R(T,(e,f+1,g))) + \gamma(S[i],'\_') \\
\quad + \gamma(S[j],T[f]); \\
//\mathrm{DELETE} \\
B(R(S,(i-1,j+1,k)),R(T,(e,f,g))) + \gamma(S[i],'\_') \\
\quad + \gamma(S[j],'\_'); \\
//\mathrm{INSERT} \\
B(R(S,(i,j,k)),R(T,(e,f+1,g))) + \gamma(T[f],'\_'); \\
\quad \begin{cases} //\text{if } k = k_0 \\ B(R(S,(i,j,k)),R(T,(e,f,g-1))); \\ //\text{else} \\ B(R(S,(i,j,k)),R(T,(e,f,g-1))) + \gamma(T[g],'\_'); \end{cases} \\
\quad \begin{cases} //\text{if } i = i_0 \\ B(R(S,(i,j,k)),R(T,(e-1,f,g))); \\ //\text{else} \\ B(R(S,(i,j,k)),R(T,(e-1,f,g))) + \gamma(T[e],'\_'). \end{cases}
\end{cases}
$$

The situation, where neither (i,j) nor (j,k) is a base pair, is also similar. The only modifications on Han's algorithm for semiglobal alignment are the additional cases for $i = i_0$ and $k = k_0$ when inserting a space on S. To fill the dynamic programming table, not all entries for all possible subranges of S need to be filled. For any given subregion $v = (i,j,k)$, a function $\zeta(v)$ can be defined as follows to determine for which subregions in S we need to fill the corresponding B entries.

$$
\zeta(v) = \begin{cases}
(i-1,j+1,k), \text{ if } (i,j) \text{ is base pair,} \\
(i,j+1,k-1), \text{ else if } (j,k) \text{ is base pair,} \\
(i-1,j,k), \text{ else if } i \text{ is single base and } i_0 \le i < x_1, \\
(i,j+1,k), \text{ else if } j \text{ is single base and } x_1 \le j < x_2, \\
(i,j,k-1), \text{ else if } k \text{ is single base and } x_2 \le k \le k_0, \\
\text{empty region, otherwise.}
\end{cases}
$$

If v defines a subregion with valid substructure, $\zeta(v)$ also defines a valid substructure. Let $v^* = (x_1 - 1, x_1, k_0)$. We only need to fill in the entries for B provided that v can be obtained from v^* by applying ζ function repeatedly. Intuitively, ζ guides which recursion formula to use. And there are only $O(m)$ such v values. In [13], they collect all these v and form a chain of triples labeled V. $(x_1 - 1, x_1, k_0)$ is the first element of the chain V and the last element of V is an empty region. Building V requires only $O(m)$ time. Then compute B for all possible subregions (e,f,g) on T from the last element of V to the first element of V. Therefore, the time complexity of the algorithm is $O(mn^3)$ and it requires $O(mn^3)$ space.

3.2 Our Method

In Han's algorithm, during the calculation of B, for each value of v on S, they need to consider all possible subregions (e,f,g) on T (where $e < f \le g$). However, for a subregion v on S, if we can estimate which (e,f,g) on T would not align with v in the optimal semiglobal alignment, then we will be able to skip those subregions and would lead to a speedup in the computation of B practically. To estimate which (e,f,g) on T would not align with a given v on S in the optimal semiglobal alignment, our method involves two steps. First we try to compute a lower bound of the optimal semiglobal alignment score (c) between S and T. That means we can compute a value c such that the optimal semiglobal alignment score between S and T will definitely greater than or equal to c. Then according the value of c, we can estimate for a given v the set of (e,f,g) which would not align with v in any of the optimal semiglobal alignments between S and T (i.e., there may be more than one alignment between S and T with same maximum score). The computation of the value of c has to be as fast as possible and the value of c should be closed to the optimal semiglobal alignment score in order to achieve a considerable speed up. Note that we guarantee that the regions skipped by our algorithm are not the answers (see Lemmas 4 and 7), thus our algorithm will not miss any answers. On the other hand, for those regions that are not skipped by our algorithm, we evaluate them in the same way as in Han's algorithm, so we also will not produce more false positives.

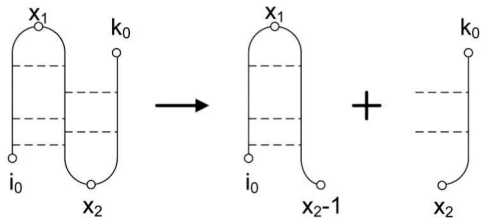

Fig. 4. Breakdown of a pseudoknot.

3.2.1 Computation of Lower Bound

Let $S[i_0 \dots k_0]$ be the query sequence with a simple pseudoknot structure M_{i_0,k_0}. As shown in the Fig. 4, a pseudoknot can be divided into two parts: 1) $S[i_0 \dots x_2 - 1]$ with a set of base pairs U; and 2) $S[x_2 \dots k_0]$ with a set of base pairs V. $U = \{(i,j) \in M_{i_0,k_0} | i_0 \le i < x_1 \le j \le x_2 - 1\}$ and $V = \{(i,j) \in M_{i_0,k_0} | x_1 \le i < x_2 \le j \le k_0\}$. According to the definition of simple pseudoknot, since $U = M_L$ and $V = M_R$ (defined in Section 2), $U \cup V = M_{i_0,k_0}$ and both of U and V are regular structures. Also, there is no branching in the structure of U (or V). That means there does not exist two base pairs $(i_1,j_1),(i_2,j_2) \in U$ (or V) such that $i_1 < j_1 < i_2 < j_2$ or $i_2 < j_2 < i_1 < j_1$.

Let $T[1 \dots n]$ be the target sequence with unknown structure. The high level idea of the heuristics to compute the lower bound of the optimal semiglobal alignment score is first to perform an optimal semiglobal structural alignment between the first part (i.e., $S[i_0 \dots x_2 - 1]$ with a set of base pairs U) and the target $T[1 \dots n]$. Let $T[e' \dots f']$ be the resulting subregions on T having optimal alignment score. We then perform an optimal prefix-global structural alignment between the second part (i.e., $S[x_2 \dots k_0]$) with a set of base pairs V and $T[f'+1 \dots n]$. The lower bound of the score is set to be the sum of the optimal semiglobal structural alignment score for the first part and the optimal prefix-global structural alignment score for the second part. The optimal prefix-global structural alignment is defined as follows.

Definition 3. *The optimal prefix-global structural alignment problem between $S[1 \dots m]$ and $T[1 \dots n]$ is to find a prefix $T[1 \dots p]$ where $0 \le p \le n$ (i.e., $T[1 \dots p]$ is an empty string when $p < 1$) such that the score of the optimal structural alignment between $S[1 \dots m]$ and the prefix $T[1 \dots p]$ is maximum. This alignment is referred as the optimal prefix-global structural alignment.*

Considering the computation of the semiglobal structural alignment between the first part $S[i_0 \dots x_2 - 1]$ with a set of base pairs U and the target $T[1 \dots n]$, due to the characteristics that there does not exist two base pairs $(i_1,j_1),(i_2,j_2) \in U$ such that $i_1 < j_1 < i_2 < j_2$ or $i_2 < j_2 < i_1 < j_1$, the time complexity of the algorithm is only $O(mn^2)$ where m is the length of S and n is the length of T. Let $Struct_U(i,j)$ be the set of base pairs $(i',j') \in U$ such that $i \le i' < j' \le j$. Define $C(i,j,e,f)$ be the score of the optimal structural alignment between $S[i \dots j]$ with substructure $Struct_U(i,j)$ and the target $T[e \dots f]$. The score of the optimal semiglobal structural alignment between $S[i_0 \dots x_2 - 1]$ and $T[1 \dots n] = \max_{1 \le e', f' \le n}\{C(i_0, x_2 - 1, e', f')\}$, where $T[e' \dots f']$ is the resulting subregions on T having optimal semiglobal structural alignment score. $C(i,j,e,f)$ can be computed as follows.

Lemma 2. *If (i,j) is a base pair, $C(i,j,e,f) = \max$*

$$\begin{cases} //\mathrm{MATCH}_{\mathrm{both}} \\ C(i+1,j-1,e+1,f-1) + \gamma(S[i],T[e]) + \gamma(S[j],T[f]) \\ \quad + \delta(S[i],S[j],T[e],T[f]); \\ //\mathrm{MATCH}_{\mathrm{single}} \\ C(i+1,j-1,e+1,f) + \gamma(S[i],T[e]) + \gamma(S[j],'\_'); \\ C(i+1,j-1,e,f-1) + \gamma(S[i],'\_') + \gamma(S[j],T[f]); \\ //\mathrm{DELETE} \\ C(i+1,j-1,e,f) + \gamma(S[i],'\_') + \gamma(S[j],'\_'); \\ //\mathrm{INSERT} \\ C(i,j,e+1,f) + \gamma(T[e],'\_'); \\ C(i,j,e,f-1) + \gamma(T[f],'\_'). \end{cases}$$

The case, which (i,j) is not a base pair, is also similar. Similarly, to fill the dynamic programming table, not all entries for all possible subranges of S needs to be filled. Let $v = (i,j)$ representing a subregion $S[i \dots j]$, a function $\xi(v)$ can be defined as follows to determine for which subregions in S we need to fill the corresponding C entries.

$$\xi(v) = \begin{cases} (i+1,j-1), \text{ if } (i,j) \text{ is base pair}, \\ (i+1,j), \text{else if } i \text{ is single base and } i_0 \le i < x_1, \\ (i,j-1), \text{else if } j \text{ is single base and } x_1 \le j < x_2, \\ \text{empty region, otherwise}. \end{cases}$$

Similarly, let $v^* = (i_0, x_2 - 1)$. We only need to fill in the entries for C provided that v can be obtained from v^* by applying ξ function repeatedly. Intuitively, ξ guides which recursion formula to use. And, there are only $O(m)$ such v values. Therefore, the total time complexity to compute the semiglobal structural alignment between the first part $S[i_0 \dots x_2 - 1]$ with a set of base pairs U and the target $T[1 \dots n]$ is $O(mn^2)$.

After obtaining the resulting subregion $T[e' \dots f']$ which has maximum semiglobal alignment score between $S[i_0 \dots x_2 - 1]$, the next step is to compute the optimal prefix-global structural alignment between the second part (i.e., $S[x_2 \dots k_0]$) with a set of base pairs V and $T[f'+1 \dots n]$ based on the resulting alignment between $S[i_0 \dots x_2 - 1]$ and $T[e' \dots f']$. Let $Struct_V(k)$ be a set of base pairs $(i,j) \in V$ such that $x_2 \le j \le k$. Define $D(k,g)$ be the score of the optimal prefix-global alignment between $S[x_2 \dots k]$ with a set of base pairs $Struct_V(k)$ and the target $T[f'+1 \dots g]$ based on the alignment between $S[i_0 \dots x_2 - 1]$ and $T[e' \dots f']$. The optimal prefix-global structural alignment between $S[x_2 \dots k_0]$ with a set of base pairs V and $T[f'+1 \dots n]$ given the alignment between $S[i_0 \dots x_2 - 1]$ and $T[e' \dots f']$ can be obtained from the entry $D(k_0,n)$. $D(k,g)$ can be computed as follows.

Lemma 3. *Let $pair(k) = j$ if $(j,k) \in V$ or -1 otherwise. Let $\tau(j) = f$ where $i_0 \le j \le x_2 - 1$ and $e' \le f \le f'$ if $S[j]$ is aligned with $T[f]$ in the resulting optimal alignment between $S[i_0 \dots x_2 - 1]$ and $T[e' \dots f']$. Otherwise, set $\tau(j) = -1$ if $S[j]$ is aligned with space in the resulting optimal alignment. $D(k,g) = \max$*

$$\begin{cases} //\text{MATCH} \\ // \text{ if } k \text{ involves a base pair in } V \text{ and } \tau(pair(k)) \neq -1 \\ D(k-1,g-1) + \gamma(S[k],T[g]) \\ \quad + \delta(S[pair(k)],S[k],T[\tau(pair(k))],T[f]); \\ // \text{ else} \\ D(k-1,g-1) + \gamma(S[k],T[g]); \\ //\text{DELETE} \\ D(k-1,g) + \gamma(S[k],'\_'); \\ //\text{INSERT} \\ //\text{if } k = k_0 \\ D(k,g-1); \\ // \text{ otherwise} \\ D(k,g-1) + \gamma(T[g],'\_'). \end{cases}$$

The time complexity of computation of the optimal prefix-global structural alignment between the second part (i.e., $S[x_2 \dots k_0]$) with a set of base pairs V and $T[f'+1 \dots n]$ based on the alignment between $S[i_0 \dots x_2 - 1]$ and $T[e' \dots f']$ can be done in $O(mn)$ time.

The lower bound of the score is set to be the sum of the optimal semiglobal structural alignment score for the first part and the prefix-global structural alignment score for the second part, which can be computed in $O(mn^2) + O(mn) = O(mn^2)$ time.

Lemma 4. *The resulting lower bound is always less than or equal to the score of the optimal semiglobal structural alignment between S and T.*

Proof. Let $T[e' \dots f']$ where $e \leq e', f' \leq f$ be the resulting region with the optimal semiglobal structural alignment with $S[1 \dots x_2 - 1]$. Let $T[f'+1 \dots g']$ where $g' \leq g$ be the resulting region with the optimal prefix-global structural alignment with $S[x_2 \dots m]$. The lower bound equals to the corresponding score of the resulting structural alignment between $S[1 \dots x_2 - 1] \cup S[x_2 \dots m]$ and $T[e' \dots f'] \cup T[f'+1 \dots g']$. Since it is one of the candidates of the optimal semiglobal structural alignment between $S[1 \dots m]$ and $T[1 \dots n]$, the resulting lower bound is always less than or equal to the score of the optimal semiglobal structural alignment between S and T. □

3.2.2 Pruning

After computing the lower bound of the optimal structural alignment between S and T, for each subregions (e,f,g) on T (where $e < f \leq g$), we will try to estimate which (e,f,g) on T would not align with v in the optimal alignment. Then subsequent steps of computation of the alignment which involves (e,f,g) aligned with v can be skipped and so it would lead to a speedup in the computation of B.

Considering the computation of semiglobal structural alignment between $S[1 \dots m]$ and $T[1 \dots n]$, for a subregion $R_x = R(S,(i,j,k))$ on S and another subregion $R_y = R(T,(e,f,g))$ on T, $B(R_x,R_y)$ is defined as the score of the optimal semiglobal structural alignment between R_x and R_y. In the following, we will compute the maximum possible alignment score between S and T provided that R_x is aligned with R_y. The situation that R_x is aligned with R_y means that $S[1 \dots i]$ is aligned with $T[1 \dots e]$ and $S[j \dots k]$ is aligned with $T[f \dots g]$. Formally, R_x is defined as aligned with R_y if for the resulting optimal alignment of S and T: $S'[1 \dots r]$ and $T'[1 \dots r]$ where $r \geq m,n$, there exists a, b, c such that after removing the spaces, $S'[1 \dots a]$ and $T'[1 \dots a]$ will be equal to $S[1 \dots i]$ and $T[1 \dots e]$ and $S'[b \dots c]$ and $T'[b \dots c]$ will be equal to $S[j \dots k]$ and $T[f \dots g]$, respectively.

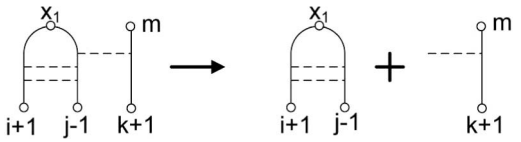

Fig. 5. Breakdown of $\neg R_x$.

Given subregions $R_x = R(S,(i,j,k))$, $R_y = R(T,(e,f,g))$ and $B(R_x,R_y)$, let $\neg R_x = S[i+1 \dots j-1] \cup S[k+1 \dots m]$ and $\neg R_y = T[e+1 \dots f-1] \cup T[g+1 \dots n]$, given R_x is aligned with R_y, when we consider the case that $i > 1$ and $k < m$, the maximum semiglobal alignment score between S and T should be equal to the sum of the optimal suffix-global alignment score between R_x and R_y and the optimal prefix-global alignment score between $\neg R_x$ and $\neg R_y$. The optimal suffix-global structural alignment is defined as follows:

Definition 4. *Optimal suffix-global structural alignment between $S[1 \dots m]$ and $T[1 \dots n]$ is to find a suffix $T[q \dots n]$ where $1 \leq q \leq n+1$ (i.e., $T[q \dots n]$ is an empty string when $q > n$) such that the score of the optimal structural alignment between $S[1 \dots m]$ and the suffix $T[q \dots n]$ is maximum.*

Let $max_s(R_x,R_y)$ be the maximum alignment score between S and T provided that R_x is aligned with R_y. Since the optimal suffix-global alignment score between R_x and R_y is always less than or equal to $B(R_x,R_y)$, if we can compute the upper bound of the optimal prefix-global alignment score between $\neg R_x$ and $\neg R_y$ (let it be $E(\neg R_x, \neg R_y)$), then $max_s(R_x,R_y)$ is less than or equal to $B(R_x,R_y) + E(\neg R_x, \neg R_y)$. If the value $B(R_x,R_y) + E(\neg R_x, \neg R_y)$ is smaller than the lower bound computed previously, we can definitely know that in any of the optimal alignments, R_x should not be aligned with R_y, and then the subsequent steps of computation of the alignments which involve R_x aligned with R_y can be skipped and so it would lead to a speedup in the computation of B.

In the following, we will provide a heuristic algorithm to compute $E(\neg R_x, \neg R_y)$. The method is time efficient and the value is proved to be greater than or equal to the optimal suffix-global alignment score between $\neg R_x$ and $\neg R_y$.

As shown in Fig. 5, the region $\neg R_x$ can be further decomposed into two parts: 1) $S[i+1 \dots j-1]$ with a set of base pairs U_{i+1}^{j-1}; and 2) $S[k+1 \dots m]$ with a set of base pairs V_{k+1}^{m}. $U_{i+1}^{j-1} = \{(i',j') \in M | i+1 \leq i' < x_1 \leq j' \leq j-1\}$ and $V_{k+1}^{m} = \{(i',j') \in M | k+1 \leq j' \leq m\}$. The upper bound of the optimal prefix-global alignment score ($E(\neg R_x, \neg R_y)$) between $\neg R_x$ (i.e., $S[i+1 \dots j-1] \cup S[k+1 \dots m]$) and $\neg R_y$ (i.e., $T[e+1 \dots f-1] \cup T[g+1 \dots n]$) is set to be the sum of 1) the optimal alignment score between $S[i+1 \dots j-1]$ with a set of base pairs U_{i+1}^{j-1} and $T[e+1 \dots f-1]$; and 2) the optimal prefix-global alignment score between $S[k+1 \dots m]$ with a set of base pairs V_{k+1}^{m} and $T[g+1 \dots n]$. Note that since the subregion R_x should have a valid substructure (i.e., there does not exist $(i',j') \in M$ such that one endpoint of (i',j') is inside R_x but another endpoint is outside R_x), consider a region $\neg R_x$, for any $(i',j') \in V_{k+1}^{m}$, $x_1 \leq i' \leq j-1$.

Let $F(i,j,e,f)$ be the optimal alignment score between $S[i\ldots j]$ with a set of base pairs U_i^j and $T[e\ldots f]$. Direct computation of $F(i,j,e,f)$ will require at least $O(n^4)$. Also, if the computation of $F(i,j,e,f)$ is required for each subregion R_x, then the total time required will be further blow up. Thus, in order to compute $F(i,j,e,f)$ efficiently, we need to make use of the triple list V in the Han's algorithm mentioned in Section 3.1. V is a chain of triples (i,j,k) which includes the subregions R_x on S we need to come across during the computation of B. There are $O(m)$ elements in the chain. V can be first computed in the beginning according to the function ζ in Section 3.1. In order words, we only need to obtain the value of $F(i,j,e,f)$ for all e,f and for those i,j such that there exists $(i',j',k') \in V$ such that $i = i'+1$ and $j = j'-1$. Thus there are only $O(m)$ number of (i,j). Let R be the set of (i,j) for which we need to compute the value of F (i.e., $R = \{(i,j) | \exists k s.t. (i-1, j+1, k) \in V\}$). The following lists out the recursive formula for the computation of $F(i,j,e,f)$:

Lemma 5. $F(i,j,e,f) = \max$

$$\begin{cases} //\text{MATCH} \\ // \text{ if } (i,j) \in U_i^j \\ F(i+1, j-1, e+1, f-1) + \gamma(S[i], T[e]) \\ + \gamma(S[j], T[f]) + \delta(S[i], S[j], T[e], T[f]); \\ // \text{ else if } (i,j') \in U_i^j \text{ where } j' < j \\ F(i, j-1, e, f-1) + \gamma(S[j], T[f]); \\ // \text{ else if } (i',j) \in U_i^j \text{ where } i' > i \\ F(i+1, j, e+1, f) + \gamma(S[i], T[e]); \\ // \text{ else (i.e., both } i \text{ and } j \text{ are single base)} \\ \begin{cases} //\text{if } (i+1,j) \in R \\ F(i+1, j, e+1, f) + \gamma(S[i], T[e]); \\ // \text{ else if } (i, j-1) \in R \\ F(i, j-1, e, f-1) + \gamma(S[j], T[f]); \\ // \text{ else (i.e., } (i+1, j-1) \in R) \\ F(i+1, j-1, e+1, f-1) + \gamma(S[i], T[e]) + \gamma(S[j], T[f]); \end{cases} \\ //\text{SINGLE} - \text{MATCH, INSERT, DELETE are similar.} \end{cases}$$

There are four situations.

1. Match for both bases (i,j) (MATCH).
2. Match for only one of bases (i,j) (SINGLE-MATCH).
3. Insertion of a base on S (INSERT).
4. Deletion of a base on S (DELETE).

When considering the MATCH situation, there are also four cases: I. (i,j) is a base pair; II. j is a single base but i forms a base pair with another base $j' < j$; III. i is a single base but j forms a base pair with another base $i' < i$; IV. Both i,j are single bases. For the case I, II, III, the formula is straightforward. For case IV, we then check which i,j such that the corresponding F has been computed. There are three possibilities, 1) $(i+1,j) \in R$; 2) $(i,j-1) \in R$; or 3) $(i+1,j-1) \in R$; For the other situations like SINGLE-MATCH, INSERT and DELETE, the formula are similar.

The following proves that for all cases, the entries we refer to during the computation of F are always valid.

Proof. When consider the situation MATCH, assume $(i,j) \in R$ (i.e., $\exists k$ s.t. $(i-1, j+1, k) \in V$), for case I, since (i,j) is a base pair, there must exist k such that $(i,j,k) \in V$. Therefore, $(i+1, j-1) \in R$. For case II, the fact that (i,j') is a base pair (where $j' < j$) implies that there exists k' such that $(i-1, j'+1, k') \in V$. Since both $(i-1, j'+1, k')$, $(i-1, j+1, k) \in V$ for some $j' < j$ and k, k', there must exist k'' such that $(i-1, j, k'') \in V$, which implies $(i, j-1) \in R$. Similarly, for case III, the fact that (i', j) is a base pair (where $i' > i$) implies that there exists k' such that $(i'-1, j+1, k') \in V$. Since both $(i'-1, j+1, k'), (i-1, j+1, k) \in V$ for some $i' > i$ and k, k', there must exist k'' such that $(i, j+1, k'') \in V$, which implies $(i+1, j) \in R$. For case IV, F is computed according to either $(i+1, j)$, $(i, j-1)$ or $(i+1, j-1)$ depending on which pair is in the R. Therefore, for all cases, during the computation of F, the entries we refer to are always valid. □

To conclude, in order to fill the dynamic programming table, not all entries for all possible subranges (i,j) of S needs to be filled For any given subregion $w = (i,j)$, a function $\Psi(v)$ can be defined as follows to determine for which subregions in S we need to fill the corresponding F entries.

$$\Psi(v) = \begin{cases} (i+1, j-1), \text{ if } (i,j) \text{ is base pair in } U_i^j, \\ (i, j-1), \text{ else if } \exists k' s.t.\ (i-1, j, k') \in V, \\ (i+1, j), \text{ else if } \exists k' s.t.\ (i, j+1, k') \in V, \\ (i+1, j-1), \text{ else if } \exists k' s.t.\ (i, j, k') \in V), \\ \text{empty region, otherwise.} \end{cases}$$

R can be obtained by using $\Psi(v)$. Thus, to compute all the F for all $(i,j,k) \in V$, the computation can be performed as preprocessing and the total time required is $O(mn^2)$.

Lemma 6. *Computation of the optimal alignment score (F) between $S[i+1\ldots j-1]$ with a set of base pairs $U(i+1)(j-1)$ and $T[e\ldots f]$ for all $1 \le e, f \le n$ and for all $(i,j,k) \in V$, requires time $O(mn^2)$.*

Similarly, the optimal prefix-global alignment score ($G(k+1, g+1)$) between $S[k+1\ldots m]$ with a set of base pairs V_{k+1}^m and $T[g+1\ldots n]$ for all k and g can be computed in time $O(mn)$. The upper bound of the optimal prefix-global alignment score between $\neg R_x$ and $\neg R_y$ (i.e., $E(\neg R_x, \neg R_y)$) is set to the sum of $F(i+1, j-1, e+1, f-1)$ and $G(k+1,\ g+1)$.

Lemma 7. *The resulting upper bound is always greater than or equal to the score of the optimal prefix-global alignment between $\neg R_x$ and $\neg R_y$.*

Proof. Consider $S[1\ldots m]$ and $T[1\ldots n]$, let $E(\neg R_x, \neg R_y)$ be the optimal prefix-global alignment score between $\neg R_x$ and $\neg R_y$ where $R_x = R(S, (i,j,k))$ and $R_y = R(T, (e,f,g))$. Since $\neg R_x = S[i+1\ldots j-1] \cup S[k+1\ldots m]$ and $\neg R_y = T[e+f\ldots f-1] \cup T[g+1\ldots n]$, let $S'[1\ldots r] \cup S''[1\ldots r']$ and $T'[1\ldots r] \cup T''[1\ldots r']$ be the resulting optimal prefix-global alignment between $\neg R_x$ and $\neg R_y$. Note that $E(\neg R_x, \neg R_y)$ is the sum of the alignment score between $S'[1\ldots r]$ and $T'[1\ldots r]$, and the alignment score between $S''[1\ldots r']$ and $T''[1\ldots r']$. Because $S'[1\ldots r]$ and $T'[1\ldots r]$ is one of the candidates when computing the optimal alignment score between $S[i+1\ldots j-1]$ and $T[e+1\ldots f-1]$, $F(i+1, j-1, e+1, f-1)$ is always greater than or equal to the alignment score between $S'[1\ldots r]$ and $T'[1\ldots r]$. Similarly, because $S''[1\ldots r']$ and $T''[1\ldots r']$ is one of the candidates when computing the optimal prefix-global alignment score between $S[k+1\ldots m]$ and $T[g+1\ldots n]$,

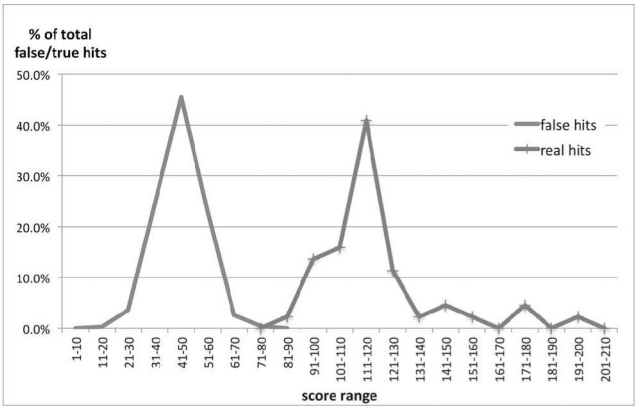

Fig. 6. Distribution of our scores between the false hits and the true hits in family RF01118.

$G(k+1, g+1)$ is always greater than or equal to the alignment score between $S''[1 \ldots r']$ and $T''[1 \ldots r']$. Therefore, the sum of $F(i+1, j-1, e+1, f-1)$ and $G(k+1, g+1)$ is always greater than or equal to the score of the optimal prefix-global alignment between $\neg R_x$ and $\neg R_y$. □

Computation of F and G are done as preprocessing. Thus, every time computation of the value of $E(\neg R_x, \neg R_y)$ which is the sum of $F(i+1, j-1, e+1, f-1)$ and $G(k+1, g+1)$ needs only constant time. If the value $B(R_x, R_y) + E(\neg R_x, \neg R_y)$ is smaller than the lower bound computed previously, we can definitely know that in any of the optimal alignments, R_x would not aligned with R_y, and then the subsequent steps of computation of the alignment which involves R_y aligned with R_x can be skipped and so it would lead to a speedup in the computation of B.

Lemma 8. *The space complexity of our method is* $O(mn^3)$.

Proof. In our method, we need to compute the tables B, C, D, E, F, and G. For each table, the number of entries we need to store during calculation is $O(mn^3)$. Thus, the overall space complexity is $O(mn^3)$. □

We remark that this space complexity is the same as that of Han's algorithm. Note also that if a region cannot be pruned, we still go through the same calculation as in Han's algorithm, thus the worst case time complexity for evaluating a region is still $O(mn^3)$, the same as that of Han's algorithm.

4 Experimental Results

We implemented our algorithm for simple pseudoknots in C++. By inputting a query RNA sequence S with its secondary structure, the program scan along the reference target sequence T and output the score for every region of T. A higher score indicates that the reference target sequence subregion is similar to the query RNA sequence S in terms of both structure and sequence. In the database Rfam 10.1, there are 71 pseudoknotted families. Among them around 40 families have simple pseudoknot structures. To evaluate the effectiveness of our algorithm, we randomly selected around 20 families with simple pseudoknot structures for the experiment. For each family, we randomly pick a seed member as the query sequence S. The target sequence T is a random sequence (with equally distribution of A, C, G, and T) of length 50,000 and the members of the family are embedded at the random positions on T.[2] Since the PAL [13] program is not available, we also implemented their method for comparison on the performance. We applied the sliding window technique for both Han's method and our method. Table 1 and Fig. 7 show the comparison of the running time between our algorithm and Han's. Our speed up gradually increases when the query length increases. The speed up is five times when the query length is 30. And the speed up is seven times when the query length is 50. The speed up is 8.5 to 11.0 times when the query length is 65 or more. The experiment was performed using a machine with 24 G memory and a quad-core 2.6 GHz CPU. It shows that our method is efficient especially for long query sequence.

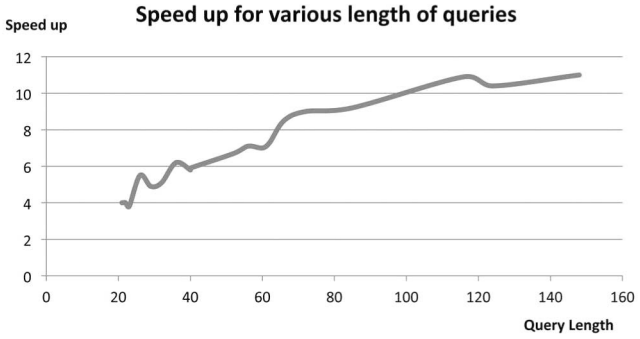

Fig. 7. Speed up for various lengths of queries.

To evaluate the efficiency of our method, we further examine the sensitivity of our algorithm and compare it with the existing method called RNATOPS [15] and BLAST. RNATOPS is a profile-based RNA structure search program that can detect RNA pseudoknots in genomes. RNATOPS is based on structures and BLAST is based on sequence, while our and Han's algorithm are based on both sequence and structure. For each family, we embedded the corresponding members into the random sequence and search it with the RNA query pattern. To understand the sensitivity of the software, say for a specific family, there are X seed members embedded and we check the top X reported regions with highest scores and see how many of them are the real members. Table 2 shows the sensitivity of our method, Han's method, RNATOPS, and BLAST. The sensitivity of our method is the same as that of Han's method. As shown in the table, the sensitivity of our method ranges from 90-100 percent with an average of 97 percent, which outperforms both RNATOPS (average 9 percent) and BLAST (average 68 percent). Fig. 6 shows an example of the distribution of our scores of the false hits (i.e., the regions which do not contain any real member) and the true hits (i.e., the regions which contain the real members) for the family RF01118. One can see that the scores of the real hits are usually higher than those of the false hits. It shows that our method could identify the real members from the target sequence.

Our algorithm consists of two preprocessing steps: 1) computation of the lower bound of the optimal structural alignment between the query sequence S and the subregion

2. If there are too many seed members in the family, we will randomly pick some of them so that the total length of all the seed members is around 2,500 (i.e., 5 percent of the total length of the target sequence).

TABLE 2
Comparison of the Sensitivity of Our Method, Han's Method, RNATOPS, and BLAST

Family ID	# of members embedded (X)	Our and Han's method			RNATOPS			BLAST		
		# of members in top X ans	sensitivity	false positive rate	# of members in top X ans	sensitivity	false positive rate	# of members in top X ans	sensitivity	false positive rate
RF00165	30	27	90%	10%	4	13%	87%	0	0%	100%
RF00390	25	24	96%	4%	2	8%	92%	25	100%	0%
RF00505	75	75	100%	0%	1	1%	99%	75	100%	0%
RF00507	23	21	91%	9%	3	13%	87%	6	26%	74%
RF01072	40	34	85%	15%	7	18%	83%	37	93%	8%
RF01074	12	12	100%	0%	0	0%	100%	2	17%	83%
RF01076	38	38	100%	0%	5	13%	87%	38	100%	0%
RF01080	16	15	94%	6%	1	6%	94%	3	19%	81%
RF01081	14	14	100%	0%	2	14%	86%	11	79%	21%
RF01087	37	35	95%	5%	2	5%	95%	3	8%	92%
RF01089	25	24	96%	4%	2	8%	92%	13	52%	48%
RF01093	38	38	100%	0%	3	8%	92%	23	61%	39%
RF01095	4	4	100%	0%	0	0%	100%	4	100%	0%
RF01097	15	15	100%	0%	2	13%	87%	15	100%	0%
RF01100	4	4	100%	0%	1	25%	75%	4	100%	0%
RF01102	8	8	100%	0%	0	0%	100%	7	88%	13%
RF01104	8	8	100%	0%	1	13%	88%	8	100%	0%
RF01109	12	12	100%	0%	2	17%	83%	12	100%	0%
RF01118	44	43	98%	2%	1	2%	98%	21	48%	52%
		Average	**97%**	**3%**		**9%**	**91%**		**68%**	**32%**

sequence on the genome T (i.e., computation of C and D); 2) preprocessing steps for pruning (i.e., computation of the matrix E). Table 3 shows the time required for each of the preprocessing steps. In summary, the total time required for both of the preprocessing steps is less than 2 percent of the total running time of our method.

To further analyze the effectiveness of our pruning method, we selected the family RF01118 and examined the percentage of table entries pruned for each subregion $R(v)$ on the query S. As shown in Fig. 8, in the beginning of the algorithm, the size of $R(v)$ is small and so it is expected that not many table entries are pruned. While the algorithm continues and the size of $R(v)$ increases, the pruning works more effectively and the number of table entries pruned increases dramatically. When the size of $R(v)$ reaches 20 percent of the query, there are over 50 percent of the table entries pruned. This statistics therefore could explain the resulting speed up of our algorithm.

TABLE 3
Breakdown of the Running Time

Family	Query length	Lower bound calculation (in second)	Pre-processing for pruning (in second)	Subtotal	% of total running time
RF00505	66	16.1	16.85	32.95	1.60%
RF00507	85	35.7	36.8	72.5	1.78%
RF01076	72	24.54	25.81	50.35	1.89%
RF01087	148	234.7	261.23	495.93	1.93%
RF01089	116	100	103.8	203.8	1.94%
RF01118	124	163.5	175.8	339.3	1.92%

5 Conclusions

We have designed an algorithm which first estimates the lower bound of the optimal alignment score and then applies an efficient pruning method for speeding up the dynamic programming algorithm for solving the RNA structural alignment problems. Experimental results show that the algorithm is effective and can achieve around 8.5 to 11 times speed up when the query length is longer than 65. On the other hand, in practice, the user sometimes would like to set a threshold so that only the subregions on the genome of which the resulting scores are higher than the threshold will be outputted. Our pruning method can work well with the inputted threshold too. The entries are then skipped if the estimated optimal alignment score for the entries is lower than the inputted threshold instead of the computed lower

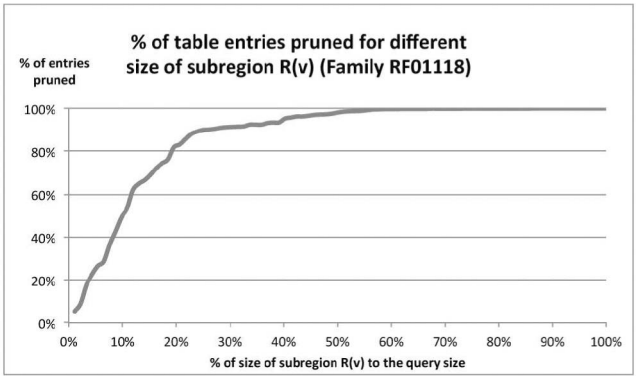

Fig. 8. Percentage of table entries pruned for different size of subregion $R(v)$ on the query S in family RF01118.

bound. By setting the threshold as the 40 percent of the maximum score for the query (i.e., the maximum score is the score when the query is aligned with the sequence exactly the same as the query), we found that the speed up can further increase to 17 times to 28 times. Although the method presented in this paper is designed for simple pseudoknots, a similar technique can also be applied for the structural alignment algorithms for other pseudoknot types.

ACKNOWLEDGMENTS

This research was supported in part by the Seed Funding Programme for Basic Research (201011159115) of HKU and by the General Research Fund (GRF) of the Hong Kong Government (HKU 719611E).

REFERENCES

[1] P.P. Gardner, J. Daub, J. Tate, B.L. Moore, I.H. Osuch, S. Griffiths-Jones, R.D. Finn, E.P. Nawrocki, D.L. Kolbe, S.R. Eddy, A. Bateman, "Rfam: Wikipedia, Clans and the Decimal Release," *Nucleic Acids Research,* vol. 39 (Database), pp. D141-D145, 2010.

[2] S.Y. Le, J.H. Chen, and J. Maizel, "Efficient Searches for Unusual Folding Regions in RNA Sequences," *Structure and Methods: Human Genome Initiative and DNA Recombination,* vol. 1, pp. 127-130, Adenine Pr, 1990.

[3] P. Clote, F. Ferré, E. Kranakis, and D. Krizanc, "Structural RNA has Lower Folding Energy than Random RNA of the Same Dinucleotide Frequency," *RNA,* vol. 11, pp. 578-591, 2005.

[4] E. Rivas and S. Eddy, "Secondary Structure Alone is Generally Not Statistically Significant for the Detection of Noncoding RNAs," *Bioinformatics,* vol. 16, no. 7, pp. 583-605, 2000.

[5] R. Klein and S. Eddy, "Research: Finding Homologs of Single Structured RNA Sequences," *BMC Bioinformatics,* vol. 4, article 44, 2003.

[6] S. Zhang, B. Hass, E. Eskin, and V. Bafna, "Searching Genomes for Noncoding RNA Using FastR," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 2, no. 4, pp. 366-379, Oct.-Dec. 2005.

[7] E.P. Nawrocki and S.R. Eddy, "Query-Dependent Banding (QDB) for Faster RNA Similarity Searches," *PLoS Computational Biology,* vol. 3, p. e56, 2007.

[8] J. Hen and C.W. Greider, "Functional Analysis of the Pseudoknot Structure in Human Telomerase RNA," *Proc. Nat'l Academy of Sciences USA,* vol. 102, no. 23, pp. 8080-8085, 2005.

[9] E. Dam, K. Pleij, and D. Draper, "Structural and Functional Aspects of RNA Pseudoknots," *Biochemistry,* vol. 31, no. 47, pp. 11665-11676, 1992.

[10] P.L. Adams, M.R. Stahley, A.B. Kosek, J. Wang, and S.A. Strobel, "Crystal Structure of a Self-Splicing Group I Intron with Both Exons," *Nature,* vol. 430, pp. 45-50, 2004.

[11] J.L. Chen and C.W. Greider, "Functional Analysis of the Pseudoknot Structure in Human Telomerase RNA," *Proc. Nat'l Academy of Sciences USA,* vol. 102, no. 23, pp. 8080-8085; discussion 8077-8079, June 2005.

[12] H. Matsui, K. Sato, and Y. Sakakibara, "Pair Stochastic Tree Adjoining Grammars for Aligning and Predicting Pseudoknot RNA Structures," *Bioinformatics,* vol. 21, pp. 2611-2617, 2005.

[13] B. Han, B. Dost, V. Bafna, and S. Zhang, "Structural Alignment of Pseudoknotted RNA," *J. Computational Biology,* vol. 15, no. 5, pp. 489-504, 2008.

[14] Y. Song, C. Liu, X. Huang, R.L. Malmberg, Y. Xu, and L. Cai, "Efficient Parameterized Algorithms for Biopolymer Structure-Sequence Alignment," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 3, no. 4, pp. 423-432, Oct.-Dec. 2006.

[15] Z. Huang, Y. Wu, J. Robertson, L. Feng, R. Malmberg, and L. Cai, "Fast and Accurate Search for Non-Coding RNA Pseudoknot Structures in Genomes," *Bioinformatics,* vol. 24, no. 20, pp. 2281-2287, 2008.

Christopher Ma received the master's degree from the Department of Computer Science at the University of Hong Kong in 2012.

Thomas K.F. Wong received the PhD degree in computer science from the University of Hong Kong in 2011. Currently, he is working as a postdoctoral fellow at the same university. His research interest includes bioinformatics.

T.W. Lam received the PhD degree in computer science from the University of Washington, in 1988. Currently, he is working as a professor at the University of Hong Kong. His research interests include bioinformatics, text indexing, and algorithms.

W.K. Hon received the PhD degree from the University of Hong Kong in 2005. Currently, he is working as an associate professor in the Department of Computer Science at National Tsing Hua University, Taiwan. He visited Purdue University from 2004 to 2006. His research interests include data compression, text indexing, and algorithm design.

K. Sadakane received the BS, MS, and PhD degrees from the Department of Information Science, University of Tokyo, in 1995, 1997, and 2000, respectively. He was a research associate in the Graduate School of Information Sciences, Tohoku University from 2000 to 2003, and an associate professor at Faculty of Information Science and Electrical Engineering, Kyushu University from 2003 to 2009. Since 2009, he has been an associate professor at the National Institute of Informatics. His research interest includes information retrieval, data structures, and data compression.

S.M. Yiu received the PhD degree in computer science from the University of Hong Kong in 1996 and is currently working as an assistant professor at the same university. His research interests include bioinformatics and computational biology. He is a member of the IEEE and the IEEE Computer Society.