Linear Time Algorithm for Calculating the Free Energy of the RNA Secondary Structure Including Pseudoknots (VS other algorithms)

Baharak Rastegari Yinglei S. Zhao Mohammad Safari
Jack Jia
Department of Computer Science
University of British Columbia
Vancouver, B.C., V6T 1Z4, Canada
{baharak,szhao,safari,jjia}@cs.ubc.ca

March 15, 2004

Abstract

We have implemented a linear algorithm proposed by condon et al.[1] to calculate the free energy of a secondary structure which includes pseudoknots by given the sequence and structure of the RNA. We integrated our work into the heuristic prediction algorithm by Ren et al.[2] (which uses a function of $O(n^2)$ for calculating the free energy) to test whether it improves the accuracy of the results and the run time of the algorithm. The results showed that in most cases it slightly improves the result and in some cases there is significant improvements in the result. But for the run time there is no improvements. We addressed some reasons of these observations in the paper and also proposed some ideas to improve our work as future work. We also compared the performance of the modified algorithm by Ren et al. [2] against that of the ILM algorithm by Zhang et al. [3]. The results showed that although their algorithm is faster than ours but our work is more accurate in predicting the base pairs in the secondary structure.

1 Motivation and Problem Description

RNA molecules have different roles in the cell. RNAs fold into structures which define their function. Therefore predicting the secondary structure of RNA is important issue and lots of work have been done on it.

Comparative sequence analysis is most reliable approach for secondary structure prediction but it needs several sequences available. So we need other approaches when just a single molecule is available. One such approach is based on the algorithms that find RNA structure with minimal free energy [9]. Secondary structure of RNA could be defined as a set of different loops and then

the free energy of the structure is calculated by summing thermodynamic free energy terms of its loops (Stacked pairs and dangling ends are considered as special types of loops).

General pseudoknotted RNA secondary structure prediction is NP-hard [10] (We will discuss about pseudoknotted structures in section 2) and therefore several polynomial complexity algorithms have been proposed to find an RNA secondary structure for a restricted class. Dynamic programming and heuristic algorithms are two widely used approaches. One of the important parts of these algorithms is finding a free energy of a secondary structure of RNA, which might be the answer (a candidate solution).

Condon et al. [1] proposed a linear time algorithm for finding a free energy of a given RNA secondary structure. In this work we will implement this algorithm and then integrate it to one prediction algorithm [2] which calculates the free energy of the partial solution (substructure) in a function. We will then compare the whole work (integrated work) with other algorithms. We also will try to compare the result with another new work by Zhang et al.[3].

In the rest of the paper we will talk about the related work in section 2 and then define the secondary structure and energy models in section 3. In section 4 we describe our algorithm and more details of implementation will be in section 5. In Section 6 we talk about the Ren et al. [2] work and integration of our algorithm into their program. Section 7 contains the data sets we use for testing the algorithm. In Section 8 we discuss about the evaluation and testing and then in section 9 we talk about the conclusion and future work.

2 Related Work

There are several dynamic programming algorithms for RNA secondary structure with pseudoknot prediction. Each of them can handle a restricted class of structures. Rivas and Eddy [3] proposed an algorithm which can handle a large class of structures and has $O(n^6)$ running time. They established an energy model that has RNA structure free energy calculation within the prediction. Their algorithm and other similar ones do not separate the energy calculation apart from the RNA secondary structure prediction instead they calculate the free energy while predicting it.

Condon et al.[1] has a linear time method for parsing a pseudoknotted structure into its elementary loops and component. We modify their parsing method and use it to calculate the free energy of a pseudoknotted secondary structure according to published sum-of-loop models.

Ren et al.[2] have implemented a heuristic approach on minimum free energy calculation. The heuristic algorithm is based on an observation that structures with very low energy are stable and more likely to form in the secondary structure. They calculate the free energy of substructures in their algorithm and have a separate function which does this task. We will mainly integrate our algorithm into their work to do a comparative evaluation of the two algorithms.

Ruan and Zhang [3] proposed an algorithm called Iterative Loop Matching (ILM) as an approach to the prediction of RNA secondary structures with pseudoknots. ILM is an extended dynamic programming algorithm that is able to predict pseudoknotted RNA secondary structures. ILM can be applied to individual sequences, using thermodynamic information to predict their structures[3].

3 Secondary Structure and Energy Model

RNA is usually single-stranded, but folds into a functional shape by forming intramolecular base pairs among some of its bases. The geometry of this pase-pairing is known as the secondary structure of the RNA. An non-pseudoknotted RNA will include hairpin Loop, multi-branched loop, internal loops, stack pairs and bulge loops (stack pairs and bulge loops are special cases of internal loops)in the secondary strucure .[6, 2]

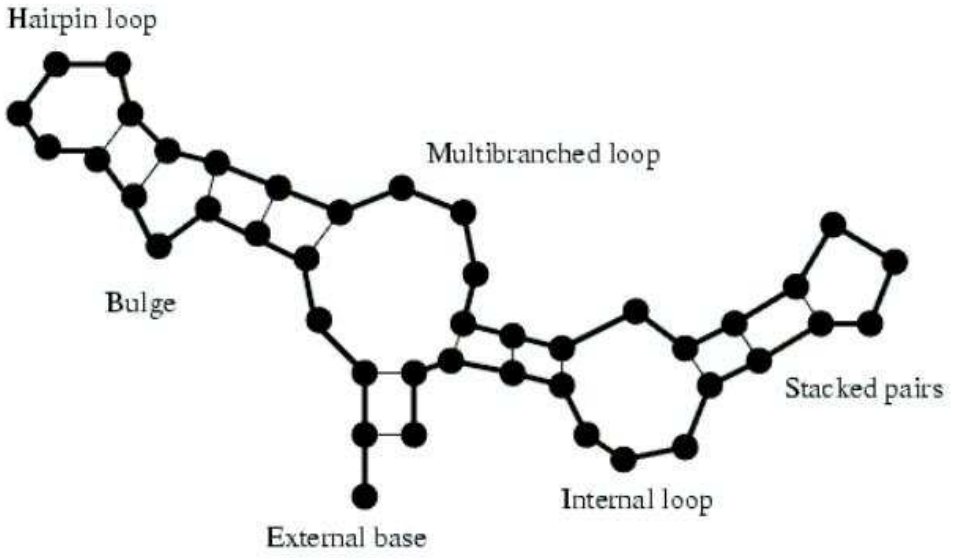

Figure 1: pseudoknot free secondary structure of an RNA strand. The thick black line indicates the backbone and the thin lines indicate paried bases

There is another special folding of RNA which is illustrated in Figure 2 and is called pseudoknot. Lets $i.j$ denotes that base i is paired with base j. We say a secondary strucure $S = s_1 s_2 ... s_n$ is a set R of base pairs $s_i.s_j$ ($1 \leq i, j \leq n$), such that each base is paired at most once. A pseudoknot in a secondary strucure S is a pair of base pairs $i.j$ and i', j' in S, with $i < i' < j < j'$.

In most MFE (minimum free energy) RNA secondary structure prediction algorithms, the free energy of the secondary structure of an RNA is calculated based on the summation of free energy of all the loops in it.

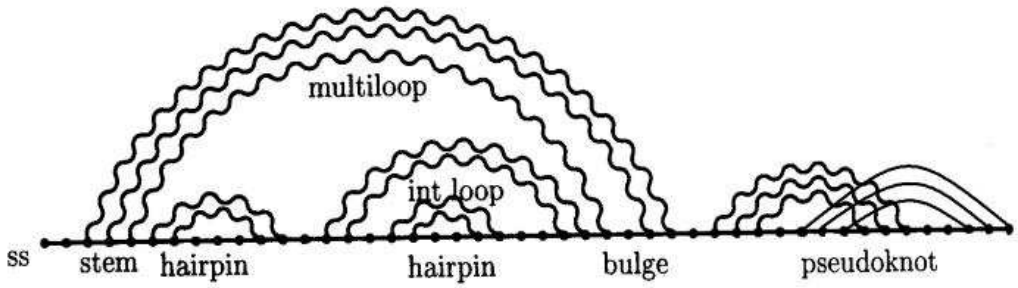

Figure 2: RNA pseudoknotted secondary structure

There is a standard model in which a function is associated with each loop type that calculates its free energy according to some specific information about it, such as its external base pairs. For a pseudoknotted structures the standard model should be extended to take pseudoknots into account. It is also possible that the free energy of other kinds of loops take different value if they have special location situation with respect to a pseudo loop.

In next subsection we describe the components of pseudoknotted structure which includes different loops and other components that are useful for calculating the free energy of the loops.

3.1 Pseudoknotted structural elements

Condon et.al [1] illustrates pseudoknotted structural elements in their work. We add some more definitions to theirs which help us in explaining the algorithm more clear. We will briefly describe all of them in the following. All of the following definition, for a fixed secondary structure R of length n, we use $[i; j]$ to denote the set of indices $i, i + 1, ..., j$ and it is a region if $i \geq j$. We get a *gapped* region by taking a union of two non-overlapping regions.

Closed Regions: We say that [i; j] is a closed region if for all base pairs $i'.j'$ of R, $i' \in [i; j]$ if and only if $j' \in [i; j]$. We refer to closed region [i; j] by $i; j$ too. Each closed region is a loop

but not all loops are closed region. Multi-pseudo-loops and Interior-pseudo-loops, which will be introduced later, are loops which are not a closed region.

Let $[i; j']$ be a closed region. If i' and j are such that $i.j$ and $i'.j'$ then we say that $i.j$ and $i'.j'$ are the *external* base pairs of $[i; j']$. If $i.j'$ then the region has just one external base pair.

Let $[i; j]$ and $[i'; j']$ be closed with $i < i'$. If $j < i'$, we say that $[i; j]$ and $[i'; j']$ are disjoint; otherwise we say that $[i'; j']$ is *nested* in $[i; j]$. In the latter case, we say that $[i'; j']$ is a child of $[i; j]$ if $[i'; j']$ is not nested in any $[i''; j'']$ with $i < i''$. We say that $[i; j]$ and $[i'; j']$ are siblings if they are children of the same closed region and $i \neq i'$. Thus the closed regions form an ordered tree $T(R)$, with one node per closed region, where the root is $[1; n]$ and children of a node are ordered by the left index of the closed region.

Pseudoknotted Regions: We say that $[i; j]$ is a pseudoknotted region of R if $i; j$ and $i.j \notin R$. We say that the indices i and j are the left and right borders of the pseudoknotted region $[i; j]$. Pair $i.j$ is pseudoknotted if there exists $i'.j'$ with $i < i' < j < j'$ or $i' < i < j' < j$.

Bands: A gapped region $[i_1; i'_1] \cup [j'_1; j_1]$ is a band of a pseudoknotted region if $i_1.j_1$ and $i'_1.j'_1$ are both pseudoknotted, $i_1 < i'_1 < j'_1 < j_1$ and there is no base pair $i''.j''$ such that : $i'' \in [i_1; i'_1] \cup [j'_1; j_1]$ and $j'' \notin [i_1; i'_1] \cup [j'_1; j_1]$; and also there is no two base pairs $i.j$ and $i'.j'$ such that (1)$i < i' < j < j'$ and (2) at least one of the following is true : (I) $i_1 < i < i'_1$ and $j'_1 < j < j_1$ (II) $i_1 < i' < i'_1$ and $j'_1 < j' < j_1$. $i.j$ and $i'.j'$ are band's closing pairs. We say that $[i_1; i'_1]$ and $[j'_1; j_1]$ are band regions. $i_1.j_1$ is outer and $i'_1.j'_1$ is inner closing pair of the band. We also say that all base pairs $i.j$ such that $i_1 \leq i \leq i'$ and $j'_1 \leq j \leq j$ are spanning the bands.

By introducing bands, each loop nested in a pseudoknot loop gets a location status as follows which will be used later in calculating the free energy of it:

- **in-Band loops:** Loops which are nested in one of the bands regions are considered to be in-Band loops. It consists of loops whose closing pairs are in a band region.
- **un-Band loops:** Lets PR denotes a pseudoknotted region. Loops which are nested in $(PR - \cup_{\forall k}([i_k; i'_k] \cup [j'_k; j_k]))$ region(where $[i_k; i'_k] \cup [j'_k; j_k]$'s are bands of a pseudoknotted region PR) are considered to be un-Band loops. It is the same to say that the closing pair of a loop is in $(PR - \cup_{\forall k}([i_k; i'_k] \cup [j'_k; j_k]))$ region.
- **span-Band loops:** loops which are nested in a pseudoknotted region, but neither are in-Band nor un-Band are considered to be span-Band loops. These kind of loops are either Multiloops which we name Multi-Pseudo loops or Interior loops which we name Interior-Pseudo loops.

Interior-Pseudo loops: An *interior loop* contains two base pairs $i.j$ and $i'.j'$ where $i < j' < j' < j$ and all bases in $[i + 1, i' - 1] \cup [j' + 1, j - 1]$ are unpaired.(bulge loops and stacked pairs are special cases of *interior loop*). IF there is a band $[bi_1; bi'_1] \cup [bj'_1; bj_1]$ such that $bi_1 < i < bi'_1$ and $bj'_1 < j < bj'_2$ (which respectively means that $bi_1 < i' < bi'_1$ and $bj'_1 < j' < bj'_2$) then it means this interior loop spans a band and is a *Interior-Pseudo loop*.

Multi-Pseudo loops : A *multiloop* contains an external base pair $i.j$ and k tuples $(i_1, j_1), (i_2, j_2), ..., (i_k, j_k)$ where $i; j, 1 \leq l \leq k$ and $i < i_1 < j_1 < i_2 < j_2 < ... < i_k < j_k < j$ and all bases in

$[i,j] - \cup[i_l, j_l], 1 \leq l \leq k$ are unpaired. If for all l, $1 \leq l \leq k$ then k should be at least 2. A *multiloop* spans a band and is named *Multi-Pseudo loop* if there is a band $[bi_1; bi'_1] \cup [bj'_1; bj_1]$ such that $bi_1 < i < bi'_1$ and $bj'_1 < j < bj'_2$) and also there is exactly one tuple (i_l, j_l) for which $i_l; j_l$ is not true (it is not a closed region) and $i_l.j_l$ spans a band ($bi_1 < i_l < bi'_1$ and $bj'_1 < j_l < bj'_2$).

Pseudoloop : Assume that $[i; j']$ be a pseudoknotted region. Let $[i_1, i'_1] \cup [j'_1, j_1] \cup [i_2, i'_2] \cup [j'_2, j_2], ..., [i_m, i'_m] \cup [j'_m, j_m]$ be bands of $[i; j]$. Let $[p_1, q_1], [p_1, q_1], ..., [p_k, q_k]$ be un-Band (un-Band closed region) children of $[i; j]$. The *pseudoloop* corresponding to [i;j] is the set $\{(i_l, j_l), (i'_l, j'_l) | 1 \leq l \leq m\} \cup \{(p_l, q_l) | 1 \leq l \leq k\}$, together with the bases in

$$[i, j'] - \cup_{l=1}^{k} [p_l, q_l] - \cup_{l=1}^{m} [i_l, i'_l] - \cup_{l=1}^{m} [j'_l, j_l]$$

to be unpaired.

3.2 Energy Models

Different algorithm extend the standard free energy model,which associates a free energy with each loop, differently.

- **Eddy and Rivas [4] (E&R) algorihtm:** The energy of the loop is calculated exactly as in the standard model if the loop is not of the kind span-Band,which means that it is neither an Interior-Pseudo loop nor a Multi-Pseudo loop. The free energy of an Interior-Pseudo loop in a standard model is multiplied by a constant g to get the free energy which is considered by $E\&R$.

  For a multi loop the energy function in standard model is generalized to $a + bu + ch + 2dm$ where a, b, c, d are constants, u is the number of unpaired bases, h is the number of tuples (i, j) in the multiloop for which $i.j$ (unpseudoknotted closed regions), and m is the number of tuples (i, j) for which $i.j \notin R$ (pseudoknotted closed region). For a Multi-Pseudo loop the constant parameters used in free energy formula are different and the energy is of the form $a' + b'u + c'h + 2d'm$.

  The energy of a pseudoloop is of the form $a + bu + cm + dk$ where a, b, c, d are constants (a is the penalty for introducing a Pseudoloop), u is the number of unpaired bases, m is the number of bands and k is the number of un-Band children.

- **Pierce's algorithm:** The energy of the loop is calculated exactly as in the standard model unless the loop is an un-Band loop. For un-Band loops different constant parameters are used.

We considered $E\&R$'s definitions in our work. But as we identify the location status for all loops and identify un-Band loops, it is easy to change our implemention to work for Pierce's definition. It could be considered as a future work to do this change and test the whole work with Pierce's definition and then compare the result with the one which uses $E\&R$'s definition.

4 Algorithm description

As we said in previoius section, there are functions that calculate the free energy of each loop. Therefore the free energy of a secondary structure for a sequence R can be calculated in linear time, given a list of its loops and some information for each one, such as: (1)an ordered list of its base pairs or tuples (2)a variable which says if a loop is of the kind span-Band or in-Band or un-Band.

So, if we could parse the structure and get all it's needed components and useful information in linear time, then we are able to calculate the free energy of the structure in linear time.

In the algorithm we keep the track of all the loops in an ordered tree $T(R)$ where as we said before the root is $[1; n]$ and children of a node are ordered by the left index of the closed region.

Then we tried to find all closed regions in one scan of the whole structure (which is stored as a sequence) and make the tree $T(R)$. Using this tree and the structure we could figure out all loops and store needed information for them.

The algorithm takes an RNA secondary structure R as input and calculates its free energy. It has different steps which briefly work as follows:

1. **Closed region finding:** In this step all closed regions of the structure are discovered and a tree $T(R)$ is created based on these closed regions. Each node (closed region) is a loop in the structure, but they are not all the loops. We will figure out other loops in another step.
2. **Loop type finding:** In this part, the algorithm decides loop type of each closed region. This is done by looking at the number (and sometimes type) of the children of each node.
3. **Band finding:** The algorithm finds the list of bands for each pseudoknotted closed region. At the end of this step an ordered list of bands regions (ordered base on the first border) is assigned to the corresponding pseudoknotted closed region. By using this ordered list it then figures out the location status of the children of pseudoknotted clsoed region. (in-Band and un-Band)
4. **Inner loop finding:** In this step the algorithm discovers Multi-Pseudo loops and Interior-Pseudo loops. It also figures out necessary information which will later be used to calculate their free energy (e.g. ordered list of tuples for Multi-Pseudo loops).

In next subsections we will describe each step more precisely.

4.1 Finding Closed Regions

Input of the algorithm is a list L of a secondary structure R. List elements are base indices, and each element points to the next and previous element in the list.Each element stores its pair index sp (0 for unpaired bases) so that we can move from one base to its paired one directly. As unpaired bases have no influence in closed region finding we would simply remove them from the list. We

will use notation $bp(a)$ for showing base pair of base a. Starting from the left we scan all indices in the list and mark the ones which are the first base of the base pair (i.e. $i < bp(i)$). There are three different types of marks:

- *B-mark:* base pairs mark. Index i is marked as B when $bp(i)$ has just been scanned.
- *P-mark:* pseudoknot mark. Index i is marked as P if there is an index i' such that $i < i' < bp(i) < bp(i')$ when $bp(i)$ has just been scanned. Furthermore, all such indices i' are also marked as P. All bases which corresponding base pair is pseudoknotted will get P mark sometime during the algorithm.
- *N-mark:* not a left border mark. Index i is marked as N if there exists i' in the same closed region such that $i' < i < bp(i') \leq l$, where l has just been scanned. Base pairs which are not the first border of a pseudoknotted region will get N mark during the algorithm.

These marks help us to find closed regions and also keeping the algorithm linear.

At each step, we check if the current element c is a right border of a closed region.

- If $bp(c)$ has only B mark then $[bp(c), c]$ is a closed region.
- If $bp(c)$ has N (and also P) mark, and the element d that previous to $bp(c)$ has both P and B mark (and not N mark) then $[d, c]$ is a pseudoknotted closed region.

In these cases we remove the closed region from the list and add a node to the tree structure.

If an index has all three marks it means that it is not a left border of a closed region and not the right border too, as if it is the latter case a closed region would be discovered. So to avoid extra works and keep the algorithm linear, we remove this index from the list.

We use a stack for facilitating the N-marking part. In this stack we keep track of the elements which haven't been N-marked yet, therefore we could avoid remarking an entry as N and keep the algorithm linear.

A tree structure T has one root which is in fact a virtual closed region. It is used to store all trees of closed region since we might have more than one tree during the execution of the algorithm (before completion).

After adding a node to the tree T we check its siblings from right to left (starting from the most recently added node) and see if the new node is its parent or not. If the new node is its parent then we make a child-parent relation between them and continue by checking next root node (else stop).

This algorithm is proved by condon et al.[1] to be linear.

4.2 Loop Type Finding

We can figure out the loop type of each closed region easily as follows.

- **Pseudoknot loop** A closed region in which the two borders are not paired with each other represents a pseudoknot loop.
- **Hairpin Loop** A non pseudoknotted closed region represents hairpin loop if it has no children.
- **Interior Loop** A non pseudoknotted closed region represents Interior loop if it has exactly one child and its child is not a pseudoknotted closed region.
- **Multi loop** A non pseudoknotted closed region which has more than one children, or it has one child which is a pseudoknot loop represents a multiloop.

4.3 Band Finding

After adding a node to the tree T and finding its type, we find band regions for it if it is a pseudoknot loop.

*L*emma 1: The only base pairs in the pseudoknotted closed region (just before we start band finding function) are the ones which span the bands.

*P*roof: Nested closed regions were removed from the list before we call the function. So the only remaining pairs are the ones which span the bands.

*L*emma 2: By the definition of bands it is easy to see that the spanning pairs in the band do not cross each other. Which means that there are not two spanning pairs $i.j$ and $i'.j'$ in a band such that either $i < i' < j < j'$ or $i' < i < j' < j$.

These two facts make the following algorithm work well for finding band borders and therefore band regions.

1. Let i be the left border of closed region
2. $(i, bp(i))$ is the outer closing pair of the band.
3. Go forward from i and backward from $bp(i)$ and check if the two new bases are paired to each other.
4. The last two bases which are paired are the inner closing pair of the band. (i' and $bp(i')$)
5. Remove the band region from the list and instead, put an entry for each region which contains the borders of them (i.e. (i, i') $(bp(i'), bp(i))$).
6. Let i be the next element after i'
7. If i is the right border of closed region stop else go to step 2

At the end of this algorithm we will have an ordered list of band regions. By using this ordered list we can easily figure out the location status of the children of the pseudoknotted clsoed region.

4.4 Inner loop Finding

We use similar approach to band finding to figure out Multi-Pseudo and Interior-Pseudo loops. We start from the outer closing pairs of each band and check all crossing pairs. Each crossing pair is either the closing pair of a Multi-Pseudo loop or Interior-Pseudo loop based on the definition of each. We should also find tuples for each Multi-Pseudo loop which is done by traversing the children of a pseudoknotted closed region. The whole approach is similar to band finding, so we won't go to more details here.

5 More Details on the Implementation

In this section we present more implementation details of what we said in previous section. As it is said before and is depicted in Fig. 3, the task of the algorithm in an abstract view is reading the input and putting it into the stack in order to find all closed regions; keeping all the loops in a tree with maintaining parent-child relationship between loops; and finally calculating the free energy for each loop and adding all these free energies up.

```
ReadInput* input = new ReadInput(len,  s, p);
Stack* stack = new Stack(input);
Bands* band = new Bands(input, st);
Loop* tree = new Loop(0, len + 1, input, band, stack);
int a, b;

for (int i = 1; i <= input->Size; i++)
  if(stack->Add(i, a, b))
      tree->addLoop(a, b);

return tree->Energy();
```

Figure 3: Main PartMain procedure of energy calculation algorithm for pseudoknotted structures

5.1 Files and Classes

Here is a description for all files and classes that we have defined.

Defines.h: contains all constants and type definitions.

Stack.h & Stack.cpp contain the implementation of class ***Stack***. This class takes a base pair, put it on the stack, and reports whenever it finds a closed region. As it is said before, it uses marking methods to efficiently do this task in linear time.

Loop.h & Loop.cpp: contain the implementation for the tree containing all closed regions. As you can see in Fig. 4 (which shows some of the properties and methods of Loop class) the tree has a right child-left sibling structure. Each loop has appropriate methods for finding its type, its nearest pseudoknotted parent and its bands if the loop is pseudoknotted. It has also

some methods for computing other values that are necessary for free energy calculation. If the loop is pseudoknotted then we have methods for finding all of its Interior-Pseudo and Multi-Pseudo loops that span its bands(span-band loops).

```
class Loop {
    int begin, end;
    LoopType type;
    Loop* RightChild, *LeftSibling, *Par-
ent, *PseudoParent;
    int NumberOfChildren;
    int NumberOfUnpaird;
    T_IntList *ILoops, *MLoops;
    Stack * St;


    void addLoop(int begin, int end);
    void FindBands();
    float getEnergy();
    void FindInnerLoops(int border1, int border2);
};
```

Figure 4: Part of Class Loop

Bands.h & Bands.cpp: finding the bands of a pseudoknotted loops are performed by calling appropriate methods of this class.

LoopList.h & LoopList.cpp: As we said before, we need to find all Interior-Pseudo and Multi-Pseudo loops that span the bands of a pseudoknotted loop. These special Interior and Multi loops are not maintained in the tree with parent-child relations. Also the energy computation for these Interior and Multi loops is different from usual ones. In a future version of our program, this class could be mixed with Loop class.

5.2 Free Energy Calculation

After finding all closed regions and putting them in the tree the calculation of the free energy is performed by traversing the tree and finding the free energy for all the loops.

As you see in Fig. 5, the free energy for each loops is calculated based on its type. Then all these values are added up together to get the free energy of the secondary structure. For unpseudoknotted loops we use functions obtained from *Mirela Andronescu*[12] for calculating free energies. These functions are for calculating the free energy of hairpin loops, interior loops and multi loops, which she has implemented for her master thesis work.

```
float Loop::getEnergy(){
    float sum = 0;
    Loop * L = RightChild;
    while (L != NULL){
            sum += L->getEnergy();
            L = L->LeftSibling;
    };
    switch (type){
        case    stackloop:
            sum += stackEnergy();
            break;
        case    hairpin :
            sum += hairpinEnergy();
            break;
        case    interior:
            sum += interiorEnergy();
            break;
        case    multi   :
            sum += multiEnergy();
            break;
        case    external:
            sum += externalEnergy();
            break;
        case    pseudo  :
            sum += pseudoEnergy();
            break;
    };

    return sum;
};
```

Figure 5: Free Energy Calculation

6 Integrating to HotKnots algorithm

6.1 HotKnots algorithm

Ren et al. [2] have implemented a heuristic approach for minimum free energy prediction of pseudoknotted RNA secondary structures. Their algorithm will be mainly used for comparison testing of our algorithm.

Ren et al. recommended a heuristic algorithm for prediction of RNA secondary structure with pseudoknots. From now on we will refer this algorithm as HotKnots algorithm. We will first introduce the algorithm and then describe how we are going to compare our method of calculating the free energy with the one utilized by Ren et al. [**?**] implementation of the algorithm.

HotKnots algorithm is also based on the minimum free energy model. In [4], a new concept called hotspot is defined as simple stem-like substructures that are comprised only of stacked pairs, bulge loops containing one unpaired base and interior loops with two (opposing) unpaired bases. These substructures are called good hotspots if they have free energy lower than -0.4kcal/mol. After

generating the initial list of all possible good hotspots, a search tree will be formed recursively. Each node of the tree represents a set of hotspots called H_n, whose size equals the depth of the node. The set of H_n will be expanded to a secondary structure using Vienna algorithm [11]. First we select good hotspots from the result of Vienna algorithm with this constraint that every base of any hotspot in H_n must remain unpaired. The set of the selected hotspots is then unioned with H_n to form a secondary structure. Note that the union of the two sets may result in constructing a pseudoknots in the structure. The free energy of the structure is then calculated, and at the end, the best predicted k structures would be the result of this method.

In HotKnots, a slightly different free energy model from Rivas and Eddy's is used. To compare this model with ours, we will substitute our method of calculating the free energy in HotKnots. For each secondary structure expanded from a node on the tree mentioned above, we can calculate the free energy with our method and compare it with the original one. Also, we will compare the efficiency of the two methods. Since for every node of the searching tree, the free energy should be calculated we are expecting to see an overall performance improvement in terms of run time by using our new method in HotKnots program.

Although HotKnots algorithm is applicable to any pseudoknotted structure, in its implementation only the free energy of structures described in Rivas and Eddy's algorithm can be calculated because of the way a structure is parsed in their approach. Since our method is more flexible, by using it HotKnots program will not have this limitation, so we may obtain better prediction.

6.2 Integration

Since the heuristic method of Ren et al.[2] does not depend on any specific energy model, we think the energy model should be a separated module in the whole program. We modified the implementation, so that a user can choose energy models and the ways the loop energy is calculated. After the modification it is straightforward now to integrate other energy calculation methods into the program. On the other hand, the method we developed can be easily integrated into other programs. We also modified the program so that any energy calculation method can have its own initialization code, and we think this will be very helpful.

Next, we will describe briefly how actually our program or any other energy calculation method can be integrated into the program. First, different method should have its own id. Second, the method should provide an interface function that given a sequence and base pairing array (i.e. A secondary structure), returns the energy of the structure. Then the function should be called in the program according to the id of the method.

Last, the code for the method should be compiled into a library and corresponding header files should be provided. It will be interesting to see more energy calculation method be integrated into the program.

7 Data sets

We have tested 25 short sequences (with known experimental structure) of length 28-86. 19 of these structure are pseudoknotted: TYMV, HDV, tRNA DA0260, tRNA DA1280, tRNA DC0010, tRNA DC0262, tRNA DD0260, tRNA DY4441, BWYV , MMTV , MMTV-vpk , SRV-1 , PKA-A , Minimal IBV, Ribozymes satRPV, tmRNA Lp_PK1 , Riboyzmes Tt-LSU_P3/P7, tmRNA Ec_PK4 , mRNA T4_gene32, mRNA Bt_PrP , mRNA Hs_PrP , mRNA Ec_S15 , mRNA Ec_alpha, mRNA T2_gene32, mRNA Ec_PK1. [2]. The test result are shown in Table 1.
We also tested some other short and long sequences[3] to get more comparable data. The testing results are in table (2).

```
procedure  Pseudospotter
input: RNA molecule S
  generate an initial list of hotspots:  h_1, h_2, ···, h_k ;
  create a tree T with a single (root) node r and empty hotspot set;
  add k child nodes to r, with the ith child having hotspot set { h, i};
  build each of the k children, as described in the next procedure;

output: list of structures sec-str(S, H_v) for all nodes v in tree T
  ranked in increasing order of free energy,
  calculated according to the model of the algorithm;

end Pseudospotter.

procedure  build
input: node v of tree T
   with hotspot set H_v
    select good hotspots that do not overlap with those in H_v;
   remove selected hotspots h if there is a node w in tree T
        with H_w = H_v∪ h;

   for each remaining selected hotspot h (if any)
      if sec-str(S, H_v ∩ h) is promising then
          create a new node v' and set H_v' to be H_v∪ h;
          recursively build up node v';
end build.
```

Figure 6: Outline of RNA secondary structure prediction algorithm

8 Evaluation and Testing

We test the algorithm on two criterias. One is the accuracy of the prediction.(table 1), and the other is the run time of each algorithm. (table 2). For evaluating the accuracy we used two functions:

- **F1 :** It is the level of accuracy which is calculated by dividing the number of base pairs that we predicted the same as in real structure (true positive) over the number of base pairs

in experimental (real) structure. So 0 means that no base pairs are predicted the same as the experimental strucure (poor result) and 1 means that all base pairs in the experimental strucure are predited (good result).

- **F2:** It is another level of accuracy which is calculated by dividing the number of base pairs we predicted but they are not in experimental structure (false positive) over the number of base pairs in experimental structure. (0 is good and 1 is poor result)

Table 1: Table for comparision of prediction results for each algorithm (1)

sequences	HotKnots algorithm		Zhang's algorithm		our algorithm	
	F1	F2	F1	F2	F1	F2
TYMV	0.84	0.12	0.64	0.68	0.84	0.12
(3rd choice)	0.96	0.04			0.96	0.04
HDV	0.933	0.1	0.967	0.433	0.633	0.267
(4th choice)					0.967	0
tRNA DA0260	0.5	0.27	0.227	1.045	0.909	0.227
tRNA DA1280	1	0.095	0.714	0.667	1	0.952
tRNA DC0010	1	0	1	0.476	1	0
tRNA DC0262	0.857	0.19	0.857	1.285	0.857	0.19
tRNA DD0260	0.571	0.619	0.571	0.619	0.286	0.714
tRNA DY4441	1	0	0.476	1.667	0.952	0.048
BWYV	1	0	1	0	1	0
MMTV	1	0.09	1	0.09	1	0.09
MMTV-vpk	1	0.09	1	0.09	1	0.09
SRV-1	1	0.09	0	1.24	1	0.09
PKA-A	1	0.083	1	0.083	1	0.083
Minimal IBV	0.647	0.294	0.941	0.235	0.941	0.118
Ribozymes satRPV	0.591	0.273	0.295	1.318	0.591	0.273
tmRNA Lp_PK1	0.8	0.2	0.5	0.2	0.8	0.2
Riboyzmes Tt-LSU_P3/P7	0.95	0.05	0.85	0.6	0.95	0.05
tmRNA Ec_PK4	1	0	0.895	0.316	0.684	0
(2nd choice)					1	0
mRNA T4_gene32	1	0	0.636	0	0.636	0.091
(2nd choice)					1	0
mRNA Bt_PrP	0.416	0.0.583	0	1.333	0.333	0.667
mRNA Hs_PrP	0.091	1.36	0.364	1.364	0.091	1.36
mRNA Ec_S15	0.588	0.353	0.941	0.588	1	0
mRNA Ec_alpha	0.458	1.125	0.5	1.917	0.458	1.083
mRNA T2_gene32	1	0	0.583	0.25	0.583	0.083
mRNA Ec_PK1	1	0	1	0	1	0

By looking at table 1 we can see that there is not much difference in $F2$ function Between HotKnots algorithm and our algorithm. But by considering $F1$ function we get that our algorithm has a better of three predictions, and HotKnots algorithm has a better of five predictions (over 25 predictions). But among our three better predicitons, two of them are significantly better than HotKnots result. One is **tRNA DA0260** and the other is **Minimal IBV**. The latter one (**Minimal IBV**) is more important as our result is the same as experimental result and we predict a pseudoknot according to the experimental structure where HotKnots algorithm doesn't find it.

The reason of difference could be that we use differenet energy functions for loops. We use Mirela's [12] function for unpseudoknotted loops and for Multi-Pseudo loops and Interior-Pseudo loops our energy model is a bit different from HotKnots (which is almost the same as in $E\&R$ paper).

Comparing Zhang's algorithm [3] and our algorithm, we see that our algorithm predicts better according to both $F1$ and $F2$ functions. According to $F2$, Zhang's [3] overall wrong prediction for each sequence is higher than ours. Considering $F1$, our algorithm predicts 10 structures better than Zhang's algorithm and Zhang's algorithm predicts 5 structures better than ours. In overall, our algorithm predicts the secondary structure more accurate than Zhang's algorithm (on tested data set).

It should be noted that compared to the experimental sequences, some of second, or third, or even fourth prediction are most close to the experimental sequences than the first prediction. In the table it is showed by mentioning $2nd$, $3rd$ or $4th$ best result after the best result for some sequences.

From table (2), we got the following observations:

Comparing to HotKnots algorithm, over 49 test sequences, our algorithm is faster on 14 of them, and HotKnots algorithm is faster on 35 of them. But by looking into each run time, the differences are small. The first conclusion is that there is not much difference on the run time between the two algorithms based on our testing results.

But we should note some points. The first is that the long sequence are the best test case for comparison of these two algorithms. We use linear algorithm for calculating the secondary structure free energy and HotKnots uses a function of $O(n^2)$ for calculating the same thing. So so for short sequences there shouldn't be that much difference and it is probable that HotKnots algorithm works better that ours. Furthermore, HotKnots calls Vienna fold function several times, which itself has cubic run time. It is probable that the run time of the whole program is dominated by the time Vienna fold function use, specially for short sequences. We have long sequences in our test set but one problem is that it seems HotKnots algorithm doesn't support sequences of length more than 256 and somehow truncates the rest of the sequence during the algorithm.

The second point is that HotKnots algorithm might does some loop finding or calculation before calling the energy calculation function which is useful for her but not for us. Right now we just replace our function, so we need to take a deeper look to the implementation to see if this is the case or not (what we didn't do because of the lack of time). If this is the case then we are doing these extra works each time which is not needed for our algorithm.

And the last one is that there is also the possibility that HotKnots algorithm uses the free energy information for secondary structures from previous steps, to calculate the free energy of new structure (by changing it a bit). We are not sure about this one too as we didn't extract all implementation issue from her work, but as each new structure is slightly different form the previous ones we think that this could be the case. If this is the case, by noting that our algorithm does the whole work form the first step for each new structure regardless of what we had before or what was the tree $T(R)$, it could cause ours to take more time. We can consider to make dynamic tree structures as future work to make the algorithm work faster.

We also tested Zhang's programs on above 49 sequences. However, since they do the prediction in two separated steps (the first step is to generate a binary matrix from the sequence, and the second step is to use the generated matrix to do the calculation) we only time their algorithm for the second step.

We consider that the run time is not a comparable criteria to our algorithms so we didn't put the results in the table. Their run time on the above 49 sequence are really fast. The time for each step of the program is only about 0.01- 4.78s.

Although Zhang's algorithm has a 18min5.26s running time for 16s sequence, but it is still not comparable since ours and HotKnots gave segmentation fault for such long sequences.

*: The sequence that got a segmentation fault by running HotKnots algorithm.

9 Conclusions and Future work

We implemented a linear algorithm which is proposed by condon et al. [1] to calculate the free energy of a secondary structure which includes pseudoknots given the sequence and structure of the RNA. To our best knowledge, this is the first 'isolated' algorithm to calculate the free energy of a given psuedoknotted RNA secondary strucure in linear time. The integration with Ren et al.[**?**] program results in more accurate prediction for some structures in the testing test, but there is not much difference in run time. (HotKnots in overall is a bit better than ours regarding run time) So we proposed some ideas as future work which might improve our work and make it fater.

We also compared the integrated work with Zhang's et al. [3] algorithm. Comparison results showed that our algorithm predicts the secondary structure more accurate than Zhang's algorithm.

Some improvements of the work can be considered for future work, such as making a dynamic tree structure instead of what we have right now to make the algorithm work faster. We can also try to optimize the modified variant of HotKnots (with our algorithm integrated in it) to decrease the run time. We can also use this algorithm and integrate it with other RNA secondary strucure prediction such as ILM [3], and also DNA secondary strucure prediction. We can also use other different energy models (such as Pierce's one) and replace them easily by what is now used in the algorithm and compare them to each other.

Acknowledgements

Here we should thank the ones who helped us, specially Mirela for her help and letting us to use her energy functions and Jihong for providing us with a heuristic prediction algorithm and also describing it to us. We should thank Holger for his ideas and help and providing the Zhang's algorithm and Anne for raising up the idea of the project.

References

[1] A. Condon, B. Davy, B. Rastegari, S. Zhao and F. Tarrant, *Understanding RNA Pseudoknotted Structures*, Unpublished manuscript.

[2] J. Ren, A. Condon and H. Hoos,*Computationally Predicting RNA Secondary Structures with Pseodoknots* , Unpublished manuscript.

[3] J. Ruan, G. D. Stormo and W. Zhang, emphIterative Loop Matching (ILM) An Approach to the Prediction of RNA secondary structures with pseudoknots, http://www.cse.wustl.edu/ zhang/projects/rna/ilm/ilm.pdf

[4] E. Rivas and S.R. Eddy (1999), emphA dynamic programming algorithm for RNA structure prediction including pseudoknots, J. Molecular Biology, 285:2053-2068

[5] E. Rivas and S.R. Eddy (2000), emphThe language of RNA: A formal grammar that includes pseudoknots, JBioinformatics, 16:334-340

[6] R.B. Lyngso and C.N. Pederson (2000), emphRNA pseudoknot prediction in energy-based models,J. Computational Biology, 7(3):409-427

[7] D.H. Mathews, J. Sabina, M. Zuker and D.H. Turner (1999), emphExpanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure, J. Mol. Biol., 288:911-940

[8] R. M. Dirks and N. A. Pierce (2003), emph A partition function algorithm for nucleic acid secondary structure including pseudoknots, J. Comput. Chem., 24(13):1664-1677.

[9] M. Zuker P. Steigler (1981), emph Optimal computer folding of large RNA sequences using thermodynamics and auxilary information ,Nucleic Acids Research, 9:133-148.

[10] R. B. Lyngso C. N. S. Pederson (2000), emph RNA pseudoknot prediction in energy based models, Journal of Computational Biology, 7(3/4):409-428.

[11] Vienna RNA secondary structure package, emphhttp://www.tbi.univie.ac.at/ ivo/RNA/.

[12] M. S. Andronescu (2003), emphAlgorithms for predicting the secondary structure of pairs and combinatorial sets of nucleic acid strands, http://www.cs.ubc.ca/grads/resources/thesis/Nov03/Mirela_Andronescu.pdf

[13] Barrette, I., G. Poisson, P. Gendron, and F. Major (2001). Pseudoknotes in prion protein mRNAs confirmed by comparative sequence analysis and pattern searching. *Nucleic Acids Res* 29(3), 753-78

[14] Ferre-D'Amare, A., K. Zhou, and J. Doudna (1998). Crystal structure of a hepatitis delta virus ribozyme. *Nature* 395(6702), 567-74.

[15] Rietveld, K., R.V. Poelgeest, C. Pleih, J. V. Boom, and L. Bosch (1982). The tRNA-like structure at the 3' teminus of turnip yello mosaic virus RNA. differnces and similarities with canonical tRNA. *SIAM J. Appl. Math.* 35(1), 68-82.

[16] Tuerk, C., S. MacDougal, and L. Gold (1992). RNA pseudoknotes that inhibit human immunodeficiency virus type 1 reverse transcriptase. *Proc Natl Acad Sci USA* 89(15), 6988-692.

[17] van Belkum, A., J. Abrahams, C. Pleih, and L. Bosch (1985). Five pseudoknotes are present at the 204 nucleotides long 3' noncoding region of tobacco mosaic virus RNA. *Nucieic Acids Res* 13(21), 7673-786.

[18] M. Tompa (2000). emphlecture notes for "Algorithms in Molecular Biology" course, http://www.cs.washington.edu/education/courses/527/00wi/

Table 2: Table for comparision of CPU time (in seconds) in RNA sequences on Columbia machine

sequences	sequence length	HotKnots algorithm	our algorithm
TYMV	86	9.70	9.19
HDV	86	7.44	6.30
tRNA DA0260	73	6.34	2.35
tRNA DA1280	71	2.20	2.53
tRNA DC0010	71	1.81	1.71
tRNA DC0262	73	2.94	4.12
tRNA DD0260	74	2.77	4.05
tRNA DY4441	70	6.64	3.13
BWYV	28	0.13	0.28
MMTV	28	0.39	0.46
MMTV-vpk	34	0.23	0.34
SRV-1	38	0.32	0.48
PKA-A	36	0.30	0.31
Minimal IBV	45	0.35	0.57
Ribozymes satRPV	73	2.13	2.73
tmRNA Lp_PK1	29	0.36	0.25
Riboyzmes Tt-LSU_P3/P7	65	1.85	1.69
tmRNA Ec_PK4	52	0.34	0.48
mRNA T4_gene32	28	0.16	0.34
mRNA Bt_PrP	45	0.81	0.59
mRNA Hs_PrP	45	0.37	0.47
mRNA Ec_S15	67	1.31	1.44
mRNA Ec_alpha	108	2.41	2.00
mRNA T2_gene32	33	0.15	0.26
mRNA Ec_PK1	30	0.17	0.23
TMV-R	105	15.36	16.27
TMV-L	84	1.75	2.48
HIVRT_I	35	0.16	0.29
Anti-HDV	32	7.19	5.96
HIVRT_I.3-2	35	0.16	0.49
HIVRT_I.3-3	35	0.24	0.29
HIVRT_I.3-6	35	0.26	0.35
HIVRT_I.3-7	35	0.26	0.29
HIVRT_I.3-22	35	0.21	0.30
HIVRT_I.3-25	35	0.14	0.26
HIVRT_I.3-50	35	0.12	0.23
5s ECEIHAOI	119	15.66	7.30
5s VS5SRRN	116	12.89	14.07
5s FVBRRAA	120	–*	19.35
srp MET.ACE	315	53.39	1:02.12
srp PYR.HOR	317	1:06.91	1:17.41
srp THE.CEL	316	2:35.45	2:24.89
telo Human	204	53.87	56.58
telo Mouse	169	38.48	46.84
telo Rat	160	36.79	42.97
telo Chicken	266	49.95	40.84
tmRNA Esch	313	1:13.65	1:11.21
tmRNA Haem	317	1:17.35	1:23.97
tmRNA Vibr	317	1:09.25	1:43.70