UNIVERSITY OF FREIBURG
GERMANY

Department Of Computer Science

Chair for Bioinformatics

Analysis and comparison of different secondary structure prediction methods for co-transcriptional RNA folding

Intersections of genomic Intervals using Interval Tree

Team Project Report

Mesbahuddin Anwari

Supervisor: Daniel Maticzka

April 2012

Abstract

Testing to find overlaps between genomic features is an important task in genomics research. We know this feature as intersection. In this project I implement a fast and flexible method to find intersections between two sets of genomic intervals by using interval trees. The implementation(unionBed) uses sets of features in BED format as input data and find overlaps between them. Then the unionBed results data is used to analyse three different secondary structure prediction hypotheses for co-transcriptional RNA folding and to compare them to each other.

Contents

Chapter 1

Introduction

The motivation of implementing unionBed was getting a faster program with which to compare two large sets of genomic intervals and to search for intersections between them. Figure 1.1 shows an example of the intersection concept. For finding overlaps between two sets of genomic intervals I should find a realistic and efficient method to compare the interval sets to each other and search for intersections. Comparing each interval of one set to all intervals of another set is an insufficient and very naive approach because it is very time consuming.

An approach to solve this problem is using interval trees. I use an interval tree to represent a set of intervals and then compare a second set of intervals to the first set by searching in the created interval tree. I used a self-balancing(or height-balanced) interval binary search tree to have an efficient search time. Every node in this type of tree structure keeps its height(path length from the root of tree to the node) small against random item insertions and deletions[6].

Then I analyse and compare three different secondary structure prediction hypotheses for co-transcriptional RNA folding. Each of these hypotheses folds sequences and calculates accessibility of the main motif(TTCTCT) from selected subsets of folding windows. To do this work I change the local folding algorithm(subsets of folding windows) and run it on the data set of local sequences. I describe the data set preparation in Chapter 3.

In this report, I start with a discussion of my choice of data representation. In the following chapter, I describe the finding of overlaps using interval trees model used for the unionBed algorithm. The smart pointers tool of the boost library [1] was used as a base for implementing this algorithm. I performed several experiments on my algorithm, the results of which are described and depicted in Chapter 2. Some details of the implementation are also explained in the same chapter. I conclude the

Figure 1.1: Graphical example of intersections between two sets of intervals which has to be detected by unoionBed program.

report with some discussion in the last chapter.

Chapter 2

Intersection Algorithm

Finding an efficient algorithm to search for intersections and overlaps between two sets of large quantities of genomic intervals is an important task. In this chapter, I explain the method I choose to find the overlaps. The main idea of my algorithm is the using of interval tree binary search. The approach is to represent the larger data set of intervals as a self height balancing interval tree(RedBlack Tree) and then use interval binary search to find the intersections of two sets. I will first discuss the format of the input data and subsequently I will explain how to build the interval tree and my unionBed algorithm.

2.1 Supported Data Format - BED Format

As explained on the UCSC Genome Browser website [2], the BED format is a flexible way to represent genomic features and genomic intervals. The BED format can describe up to 12 columns, but only the first 3 columns and column 6 for any comparisons of intervals would be needed. I use also columns 4 and 5 of the BED format in implementation of the interval tree to have nodes with specific properties. These properties can be helpful during the creation of the interval tree to prevent duplicates. Input BED files should be TAB separated.

1. chrom - The name of the chromosome of genomic feature.
2. start - The starting position of the feature in the chromosome.
3. end - The ending position of the feature in the chromosome.
4. name - Defines the name of the BED feature.

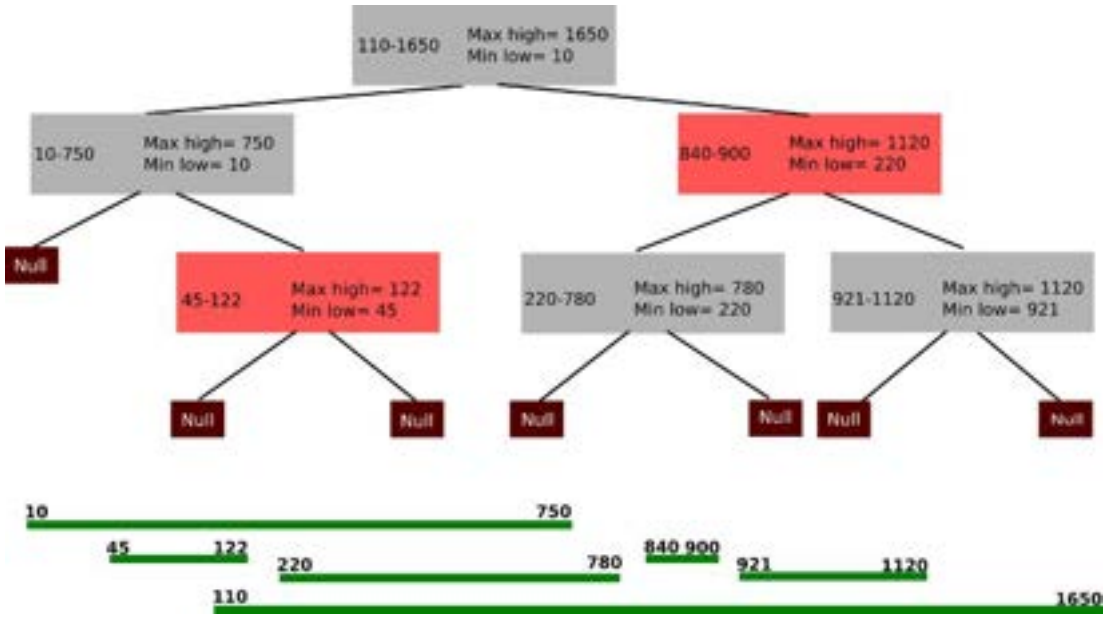
5. score - Defines score of the BED feature.
6. strand - The strand of BED feature- '+' or '-'.

For example: chr19 11823 14433 ucChr19cc 0 -

2.2 Interval Trees

In informatics, an option to represent intervals is an interval tree which is a binary tree data structure. It is an efficient data structure, especially when we are searching to find intervals that overlap with all intervals of another data set. Another useful application of interval trees is windowing requests. One example is finding all intersections of two large data sets. A similar data structure is the segment tree.

The naive solution is to compare each interval and check whether it overlaps the input interval, which running time is $O(nm)$, where n is the number of intervals in the first set and m is the number of intervals in the second set. Since a query may return all intervals, for example if the query is a large interval overlapping all intervals in the data set, this naive approach could be optimal. Interval trees are dynamic, i.e., they allow insertion and deletion of intervals. They obtain a query time of $O(logn)$, while the preprocessing time to construct the data structure is $O(nlogn)$ (but the space consumption is $O(n)$)[5].



Graphic example of the interval tree's structure.

2.2.1 Red-Black Tree

One of the self height balancing binary tree data structures is Red-Black tree. The tree operations such as deletion, insertion and search in the tree have an efficient running time $O(logn)$ where n is the number of nodes in the tree. In Red-Black tree each node has a color red or black, which it is used to have a balance tree.

2.3 unionBed Program - Algorithm

One important task applied on two sets of genomic intervals is to find overlaps between intervals of one set with another set of intervals. This task is called intersection. Efficiency and time consumption are the problems we have to consider when the algorithm should process large input data sets. In this research unionBed program has large BED files of genomic interval features as input and it should find all intersections between these intervals. In the sub-chapter below, some of the implementation details of my model of unionBed program is being discussed. I will also introduce chosen values for some of the system parameters.

2.3.1 Usage and Option details

Usage: unionBed -a <bed filename> -b <bed filename> [OPTIONS]

Options description:

-of <filename> Print results to this file.

-s Require same strandedness, only report overlaps on the same strand.

-n Minimum number of overlapping nucleotides.

An example:

$ cat BedFile1.bed
chr1 150 300
chr1 800 1200

$ cat BedFile2.bed
chr1 1200 280

$ unionBed a BedFile1.bed b BedFile2.bed
chr1 150 280

2.3.2 Implementation of unionBed Program

The implementation of the algorithm is based on C++. I use the boost library [1] for some of the features needed for my model, namely smart pointers tool and list feature of pointer container tool. Boost is an open source C++ library with boost Software License, which it is free of charge, to anyone provision a copy of the software. I use smart pointers tool from boost library for the interval tree implementation to handle allocated memory dynamically and efficiently.

2.3.3 Interval Trees of Genomic Features

As input of my program I get two files of genomic intervals in BED format. In order to create the interval trees, I need the first 6 columns of each interval in BED format from the input data set. Each genomic interval is represented by a node in the tree. We make a multiple tree(forest) for the larger input BED file, one tree for each chromosome set. Each tree represents intervals of one chromosome from the input data set.

2.3.4 Searching and Finding all Intersections

The preprocessing time to construct the multiple interval tree for the input set of genomic intervals is $O(nlogn)$, which n is the number of the intervals of the larger input set file. In the first step unionBed program finds the larger file between two input files of intervals to create the interval tree. The search operation in interval tree data structure has running time of $O(logn)$. If we have m features in the second input set file and searching for all overlaps between these features in the constructed multiple tree, then running time for finding intersections should be $O(mlogn)$. Overall time consumption by unionBed program is $O(nlogn) + O(mlogn) = O(n + mlogn)$, which $O(nlogn)$ is the time to make the interval tree and $O(mlogn)$ is the time to search and find all overlaps of intervals of the second input file in the tree.

Chapter 3

Analysis of Prediction Methods

Analysis and comparison of different secondary structure prediction methods for co-transcriptional RNA folding will be discussed in this chapter. First of all I explain about folding algorithm and then the data set, at the end I analyse the results.

3.1 Folding Algorithm

The local-fold algorithm was changed to fold and calculate the accessibilities of the motifs in the sequences. The local-fold algorithm used Perl wrapper for RNAplfold from Vienna package[3] and it can be used for different purposes of folding local sequences, but I need it to calculate the probabilities of secondary structures and compare the results. The parameters of folding program are set to specific values, the window size(W) was set to 150 and span size(compute only basepairs with maximal span L) was set to 100.

3.2 Data Preparation

In preparation for analysis of my 3 hypotheses of PTB(Polypyrimidine Tract-binding Protein) with the main motif TTCTCT, I used the data set of the PTB paper[7]. The raw data from this paper was already mapped to the human genome(hg18), this results in genomic coordinates. As described in the paper, they also clustered PTB binding events and identify peaks above the gene-specific. Then the resulting peaks were merged to clusters by placing PTB peaks within a 50nt window. In the first step, I find all TTCTCT motifs in human genome and intersect them to the PTB data(the peaks), then intersect the results to introns(RefSeqIntrons) by using the

intersection program(unionBed) for positive data and sequences, with data set size of 6772 (the number of motifs) sequences. The overlapped motifs between the data set and human genome provide the positive data. Next step is to prepare the result sequences for folding by adding 200nt to each side of the main motif which is called slopping. To do this task I used the slopBed program from BedTools[4].

To prepare the negatives I found non-overlaps between the raw PTB data set and the human genome(hg18) with the main motif TTCTCT and then intersected the results to introns by using my intersection program(unionBed). The result data set has size of 1338080 sequences, that is the number of TTCTCT motifs not overlapping positive sequences. To get the same size of positives data, I randomly choosed the number 6772 sequences from the negatives data and prepare the result sequences for folding by slopping them to 200nt. Figure 3.1 shows a graphical example of preparation positives data with the main motif TTCTCT for folding and calculation of accessibilities. For the negatives I carry out the same process except that instead of finding overlaps step, I look for non-overlaps.

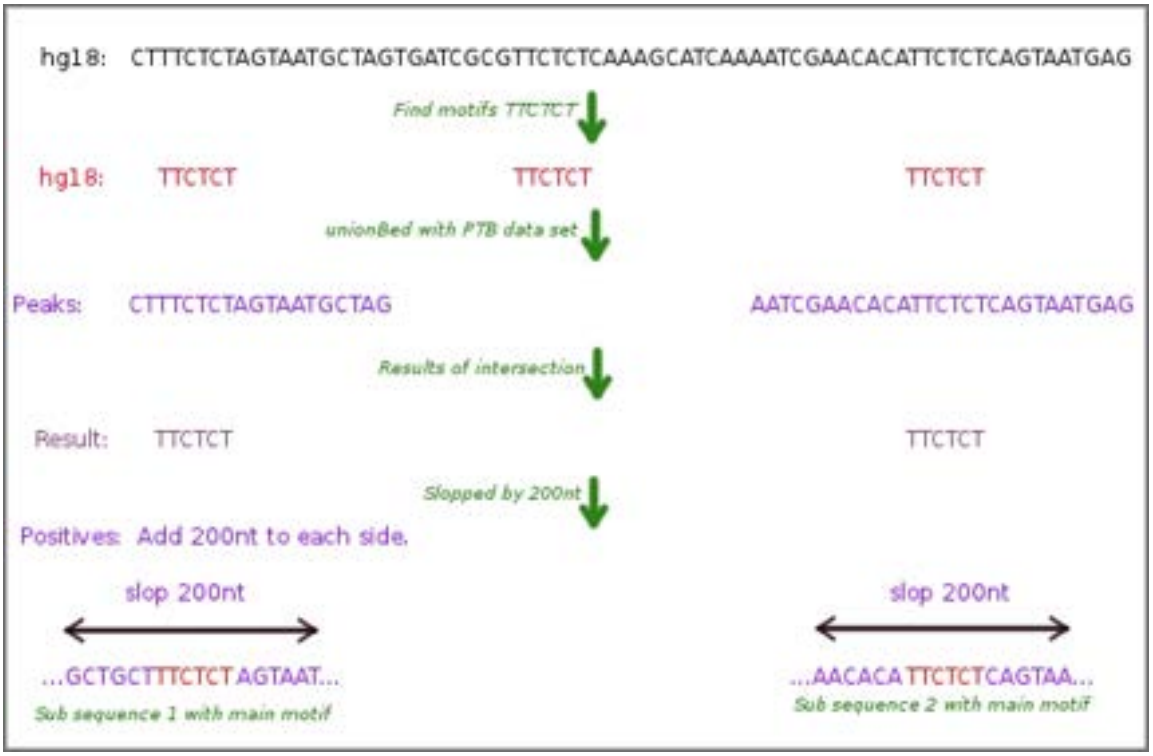


Figure 3.1: Graphical simulation of preparation positives data.

3.3 Folding Hypotheses

I consider three hypotheses to calculate the accessibilities of the main motif TTCTCT in the data set sequences by using local-fold folding algorithm with parameters $W = 150$ and $L = 100$. According to the average pair probabilities over windows of size $W = 150$, I subtract the motif size which is 6 from window size($W = 150$). We divide the result(size of 144) by three to get 40nt for applying on each hypotheses. I choose 40nt for each hypothesis, because three times 40nt is 120nt, if we subtract it from 144nt, remains 24nt which we can not include it to any of the hypotheses.

These hypotheses are as follows:

1. Hypothesis A: Using the first 40 windows overlapping motif at 5' end(co-transcriptional).

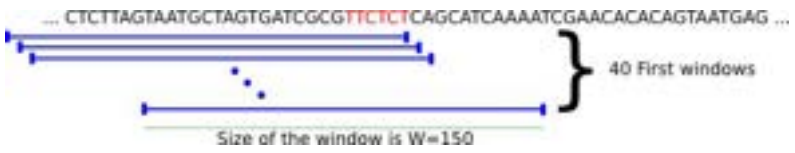


Figure 3.2: Windows used for Hypothesis A.

2. Hypothesis B: Using the last 40 windows overlapping motif at 3' end(anti co-transcriptional).

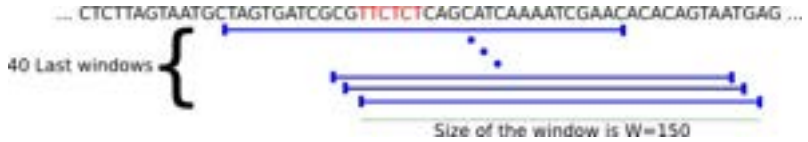


Figure 3.3: Windows used for Hypothesis B.

3. Hypothesis C: And using the 40 windows at the middle(neutral).

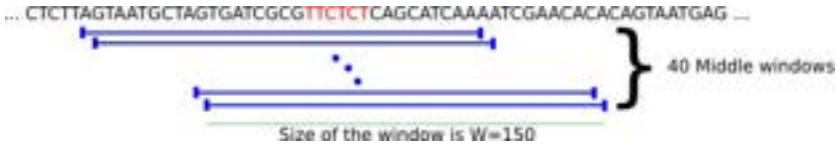


Figure 3.4: Windows used for Hypothesis C.

3.4 Results

After preparation of the data and sequences, I use the modified local-fold program to apply on data set and calculating the accessibilities of TTCTCT motif from 40 windows selected for the respective hypothesis in sequences and compare three hypotheses to each other. You can see the results in plots as follows.

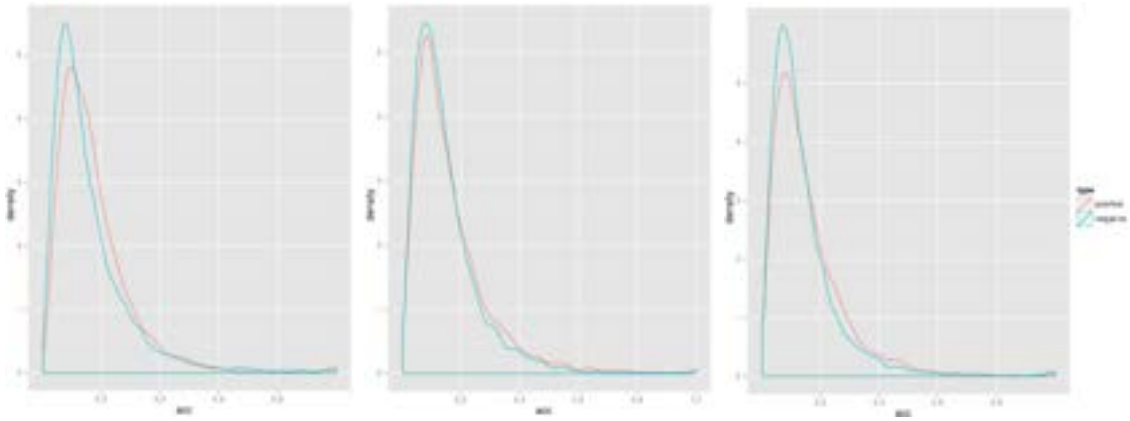


Figure 3.5: Results plot of hypotheses A(left), B(middle) and C(right).

You can see that hypothesis A gives a better results and it shows the motifs in positives data are more accessible.

Chapter 4

Discussion

In this work I implemented a program which finds intersections of genomic intervals using Interval Trees. I used a Red-Black tree(interval tree) data structure for implementing the unionBed program. The Red-Black tree data structure(self height balancing tree) was chosen in order to attain faster runtime during the searching process to find overlaps. The searching and finding intersections of genomic intervals was done with a running time $O(mlogn)$. One of the crucial issues in implementing the unionBed program was the choice of using pointers for interval tree data structure. I used smart pointers from Boost library for this purpose. Next I discussed an analysis and comparison of different secondary structure prediction hypotheses for co-transcriptional RNA folding. Using a modified version of the local-fold folding algorithm to calculate the accessibility of TTCTCT motif from 40 windows for each hypothesis. The comparison between three different secondary structure prediction hypotheses shows that hypothesis A(first 40 windows) works best. The reason for this is that the hypothesis A folds sequence windows for the sequence parts that are created first at transcription and the motifs are more accessible.

Acknowledgements

I would like to thank my supervisor Daniel for his patience and guidance through this work.

Bibliography

[1] Boost c++ librairies. http://www.boost.org.

[2] Ucsc genome browser website, bed format. http://genome.ucsc.edu/FAQ/FAQformat.html#format1.

[3] Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

[4] Aaron R. Quinlan and Ira M. Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, March 2010.

[5] Wikipedia. The free, collaborative, multilingual internet encyclopedia, interval trees. http://en.wikipedia.org/wiki/Interval_tree.

[6] Wikipedia. The free, collaborative, multilingual internet encyclopedia, self balancing binary search tree. http://en.wikipedia.org/wiki/Self-balancing_binary_search_tree.

[7] Yuanchao Xue, Yu Zhou, Tongbin Wu, Tuo Zhu, Xiong Ji, Young-Soo Kwon, Chao Zhang, Gene Yeo, Douglas L Black, Hui Sun, and et al. Genome-wide analysis of ptb-rna interactions reveals a strategy used by the general splicing repressor to modulate exon inclusion or skipping. *Molecular Cell*, 36(6):996–1006, 2009.