ABSTRACT

ALGORITHMS FOR PRE-MICRORNA CLASSIFICATION AND A GPU PROGRAM FOR WHOLE GENOME COMPARISON

by
Ling Zhong

MicroRNAs (miRNAs) are non-coding RNAs with approximately 22 nucleotides that are derived from precursor molecules. These precursor molecules or pre-miRNAs often fold into stem-loop hairpin structures. However, a large number of sequences with pre-miRNA-like hairpin can be found in genomes. It is a challenge to distinguish the real pre-miRNAs from other hairpin sequences with similar stem-loops (referred to as pseudo pre-miRNAs). The first part of this dissertation presents a new method, called MirID, for identifying and classifying microRNA precursors. MirID is comprised of three steps. Initially, a combinatorial feature mining algorithm is developed to identify suitable feature sets. Then, the feature sets are used to train support vector machines to obtain classification models, based on which classifier ensemble is constructed. Finally, an AdaBoost algorithm is adopted to further enhance the accuracy of the classifier ensemble. Experimental results on a variety of species demonstrate the good performance of the proposed approach, and its superiority over existing methods.

In the second part of this dissertation, A GPU (Graphics Processing Unit) program is developed for whole genome comparison. The goal for the research is to identify the commonalities and differences of two genomes from closely related organisms, via multiple sequencing alignments by using a seed and extend technique to choose reliable subsets of exact or near exact matches, which are called anchors. A

rigorous method named Smith-Waterman search is applied for the anchor seeking, but takes days and months to map millions of bases for mammalian genome sequences. With GPU programming, which is designed to run in parallel hundreds of short functions called threads, up to 100X speed up is achieved over similar CPU executions.

ALGORITHMS FOR PRE-MICRORNA CLASSIFICATION AND A GPU PROGRAM FOR WHOLE GENOME COMPARISON

by
Ling Zhong

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

August 2016

APPROVAL PAGE

ALGORITHMS FOR PRE-MICRORNA CLASSIFICATION AND A GPU PROGRAM FOR WHOLE GENOME COMPARISON

Ling Zhong

Dr. Usman Roshan, Dissertation Advisor Associate Professor, Department of Computer Science, NJIT	Date
Dr. Zhi Wei, Committee Member Associate Professor, Department of Computer Science, NJIT	Date
Dr. James McHugh, Committee Member Professor, Department of Computer Science, NJIT	Date
Dr. Andrew Sohn, Committee Member Associate Professor, Department of Computer Science, NJIT	Date
Dr. Songhua Xu, Committee Member Assistant Professor, Department of Information Systems, NJIT	Date

BIOGRAPHICAL SKETCH

Author:	Ling Zhong
Degree:	Doctor of Philosophy
Date:	August 2016

Undergraduate and Graduate Education

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2016
- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2009
- Bachelor of Engineering in Flying Vehicle Design and Applied Mechanics,
  Beijing University of Aeronautics and Astronautics, Beijing,
  People's Republic of China, 2001

Major: Computer Science

Presentations and Publications:

Ling Zhong, Junilda Spirollari, Jason T. L. Wang and Dongrong Wen. (2014) "RNA Classification and Structure Prediction: Algorithms and Case Studies," *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, (eds. Mourad Elloumi and Albert Y. Zomaya), Chapter 31, (pp. 685-702). Hoboken, New Jersey: John Wiley & Sons, Inc.

Ling Zhong, Jason T. L. Wang, Dongrong Wen, Virginie Aris, Patricia Soteropoulos, and Bruce A. Shapiro. (2013) "Effective Classification of MicroRNA Precursors Using Feature Mining and AdaBoost Algorithms." *OMICS: A Journal of Integrative Biology* **17**(9): 486-493.

Ling Zhong, Jason T. L. Wang, Dongrong Wen, and Bruce A. Shapiro. (2012). "Pre-miRNA Classification via Combinatorial Feature Mining and Boosting." *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine*: 369-372.

Michael Bao, Miguel Cervantes-Cervantes, Ling Zhong and Jason T. L. Wang. (2012). "Searching for Non-coding RNAs in Genomic Sequences Using ncRNAscout." *Genomics, Proteomics & Bioinformatics* **10**(2): 114-121.

TABLE OF CONTENTS

Chapter **Page**

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This dissertation investigates several data mining problems that arise in the computational systems biology field. In the first part of the dissertation, a new approach for microRNA classification is presented. MicroRNAs (miRNAs) are non-coding RNAs with approximately 22 nucleotides that are derived from precursor molecules. These precursor molecules or pre-miRNAs often fold into stem-loop hairpin structures. However, a large number of sequences with pre-miRNA-like hairpins can be found in genomes. It is a challenge to distinguish the real pre-miRNAs from other hairpin sequences with similar stem-loops (referred to as pseudo pre-miRNAs). Several computational methods have been developed to tackle this challenge. This dissertation presents a new method, called MirID, for identifying and classifying microRNA precursors. Seventy-four features from the sequences and secondary structures of pre-miRNAs were collected; some of these features were taken from previous studies on non-coding RNA prediction while others were suggested in the RNA folding literature. MirID is comprised of three steps. Initially, a combinatorial feature mining algorithm is developed to identify suitable feature sets. Then, the feature sets are used to train support vector machines to obtain classification models, based on which classifier ensemble is constructed. Finally, an AdaBoost algorithm is adopted to further enhance the accuracy of the classifier ensemble. Experimental results on a variety of species demonstrate the good performance of the proposed approach, and its superiority over existing methods.

In the second part of this dissertation, we present our effort to develop a GPU (Graphics Processing Unit) program for whole genome comparison. The goal for the research is to identify the commonalities and differences of two genomes from closely related organisms, via multiple sequencing alignments by using a seed and extend technique to choose reliable subsets of exact or near exact matches, which are called anchors. A rigorous method named Smith-Waterman search will be applied for the anchor seeking, but it takes days and months to map millions of bases for mammalian genome sequences. With GPU programming, which is designed to run in parallel hundreds of short functions called threads, algorithm running can achieve up to 100X speed up over similar CPU executions.

CHAPTER 2

EFFECTIVE CLASSIFICATION OF MICRORNA PRECURSURS USING FEATURE MINING AND ADABOOST ALGORITHM

2.1 Background

MicroRNAs (miRNAs) are non-coding RNAs (ncRNAs) of approximately 22 nucleotides that are known to regulate post-transcriptional expression of protein-coding genes (Bartel 2004, Bindra, Wang et al. 2010). Lee et al. (Lee, Feinbaum et al. 1993) first reported that in *C. elegans*, lin-4 regulates the translation of lin-14 mRNA via an antisense RNA-RNA interaction. Since then, many functions of miRNAs have been discovered (Aukerman and Sakai 2003, Brennecke, Hipfner et al. 2003, Johnston and Hobert 2003, Bushati and Cohen 2007, Mack 2007). They have been shown to play a very important role in the transcriptional and post-transcriptional regulation of genes affecting protein levels. They can have multiple mRNA targets as they bind to the targets with partial complementarities in animals. In addition, the mRNA targets can be regulated by multiple microRNAs. They are likely involved in regulation of all biological processes, and are also found circulating in blood (Mitchell, Parkin et al. 2008, Scholer, Langer et al. 2011). Their expression has been shown to be correlated with the expression of oncogenes in cancer cells (Sampson, Rong et al. 2007, Zhu, Wu et al. 2008), cancer risk factors (Wang, Zhang et al. 2007) and drug metabolism (Tsuchiya, Nakajima et al. 2006, Takagi, Nakajima et al. 2008, Gomez and Ingelman-Sundberg 2009, Pan, Gao et al. 2009). They hold a great potential for pharmacogenomics applications, such as the tailoring of drugs to the specific cancers and monitoring the response to, and toxicity of, the drugs in individual patients.

MiRNAs are derived from pre-miRNAs that often fold into stem-loop hairpin structures. These characteristic stem-loop structures are highly conserved in different species (Lai, Tomancak et al. 2003). One challenging research problem is to distinguish pre-miRNAs from other sequences with similar stem-loop structures (referred to as pseudo pre-miRNAs). Many computational methods have been developed to tackle this challenge. A common approach is to transform the classification of real and pseudo pre-miRNAs to a feature selection problem.

Lim et al. (Lim, Glasner et al. 2003) reported some characteristic features in phylogenetically conserved stem loop pre-miRNAs. Lai et al. (Lai, Tomancak et al. 2003) considered hairpin structures predicted by mfold (Zuker 2003) as well as the nucleotide divergence of pre-miRNAs. Xue et al. (Xue, Li et al. 2005) decomposed stem-loop hairpin structures into local structure-sequence features, and used these features in combination with a support vector machine to classify pre-miRNAs. Bentwich et al. (Bentwich, Avniel et al. 2005) proposed a scoring function for pre-miRNAs with thermodynamic stability and certain structural features, which capture the global properties of the hairpin structures in the pre-miRNAs. Ng and Mishra (Ng and Mishra 2007) employed a Gaussian radial basis function kernel as a similarity measure for 29 global and intrinsic hairpin folding attributes, and characterized pre-miRNAs based on their dinucleotide subsequences, hairpin folding, non-linear statistical thermodynamics and topology. Huang et al. (Huang, Fan et al. 2007) evaluated features valuable for pre-miRNA classification, such as the local secondary structure differences of the stem regions of real pre-miRNA and pseudo pre-miRNA hairpins, and established correlations between different types of mutations and the secondary structures of real pre-miRNAs.

More recently, Zhao et al. (Zhao, Wang et al. 2010) considered structure-sequence features and minimum free energy of RNA secondary structure, along with the double helix structure with free nucleotides and base-pairing features. In general, the quality of selected features directly affects the classification accuracy achieved by a method.

In this dissertation, we present a combinatorial feature mining method for pre-miRNA classification. Our method, named MirID, identifies and classifies an input RNA sequence as a pre-miRNA or not. MirID considers different combinations of features extracted from pre-miRNAs. For each combination (or each set of features), we create a support vector machine (SVM) model (Cortes and Vapnik 1995, Fan, Chen et al. 2005) based on that feature set. SVM models whose accuracies are above a user-determined threshold are then used to build a classifier ensemble. This classifier ensemble will be refined through several iterations until its accuracy cannot be enhanced further. Next, we construct new feature sets based on the best feature sets obtained so far by performing pairwise merge and split operations on the best feature sets. Then, we repeat the above procedure iteratively by building a SVM model based on each new feature set, constructing a classifier ensemble from the SVM models whose accuracies are above the newly computed threshold, and refining the ensemble until it cannot be improved further. Finally, we output the best classifier ensemble obtained through this iterative procedure. To further enhance the accuracy of the classifier ensemble, we apply a boosting algorithm to the ensemble to obtain a strong classifier, which is used for pre-miRNA classification.

The study reported here extends our previous work (Zhong, Wang et al. 2012) where we sketched the algorithms utilized by MirID. The extensions include (1) a detailed description of the MirID algorithms with a complete flowchart; (2) a larger data set containing twenty one species, as opposed to eleven species considered in (Zhong, Wang et al. 2012), with new sequences; (3) new feature values mined from these new sequences and hence, new (SVM) classification models obtained from the new data; (4) a thorough experimental study for evaluating the performance and behavior of the MirID algorithms; (5) a web server for online access as well as a downloadable tool for local use; and (6) discussion of potential applications of the software in genomics and medicine.

2.2 Materials and Methods

2.2.1 Datasets

We collected real pre-miRNAs and pseudo pre-miRNAs from 21 species, some of which were studied previously while others have not been explored. This collection is comprehensive, covering a wide variety of species, from viruses to humans. The RNA sequences were evenly divided into training data and test data. Table 2.1 presents a summary of the data. The first column of Table 2.1 shows a species or organism name. The second column of Table 2.1 shows the number of training sequences followed by the number of test sequences with respect to the organism’s real pre-miRNAs. The third column of Table 2.1 shows the number of training sequences followed by the number of test sequences with respect to the organism’s pseudo pre-miRNAs. As an example, referring to *Arabidopsis thaliana* in Table 2.1, its training set contains 66 real pre-miRNAs and 923 pseudo pre-miRNAs; its test set contains 67 real pre-miRNAs and 924 pseudo pre-miRNAs.

Table 2.1 Summary of Datasets

Species	Real pre-miRNA	Pseudo pre-miRNA
Arabidopsis thaliana	66, 67	923, 924
Caenorhabditis briggsae	66, 67	437, 438
Caenorhabditis elegans	84, 85	595, 596
Canis familiaris	161, 161	904, 905
Ciona intestinalis	160, 160	733, 734
Danio rerio	170, 170	1071, 1072
Drosophila melanogaster	81, 82	694, 694
Drosophila pseudoobscura	98, 99	495, 495
Epstein barr virus	12, 13	119, 119
Gallus gallus	241, 241	1186, 1186
Homo sapiens	504, 504	1999, 2000
Macaca mulatta	222, 223	1086, 1086
Medicago truncatula	111, 111	116, 116
Mus musculus	315, 315	2019, 2019
Oryza sativa	172, 172	522, 523
Physcomitrella patens	73, 74	703, 704
Populus trichocarpa	94, 95	809, 810
Pristionchus pacificus	60, 61	58, 58
Rattus norvegicus	193, 193	1238, 1238
Schmidtea mediterranea	72, 73	201, 202
Taeniopygia guttata	94, 95	483, 483

The real pre-miRNAs were downloaded from miRBase available at http://www.mirbase.org/ (Kozomara and Griffiths-Jones 2011). We used RNAfold (Hofacker 2003) to predict the secondary structures of all the RNA sequences. The lengths of the real pre-miRNAs in the final dataset ranged from 60 to 120 nt. The pseudo pre-miRNAs used in this study were collected from GenBank (http://www.ncbi.nlm.nih.gov/genbank/). As in (Xue, Li et al. 2005), we searched for the protein-coding regions of the genome sequences of the twenty one species in Table 2.1, and divided the regions into short sequences. The lengths of these short sequences were randomly generated, ranging from 60 to 120 nt. The pseudo pre-miRNAs were chosen from these short sequences. The criteria used in choosing the pseudo pre-miRNAs are: (i) they have a stem-loop hairpin structure, (ii) they contain at least 18 base pairs, including Watson-Crick and wobble base pairs, on the stem region of the stem-loop structure, and (iii) their secondary structure has a maximum of -15 kcal/mol free energy without multiple hairpin loops (Kozomara and Griffiths-Jones 2011). These criteria ensure that the secondary structures of the pseudo pre-miRNAs are similar to those of the real pre-miRNAs.

2.2.2 Feature Pool

In designing our pre-miRNA classification method, we examined multiple features extracted from a pre-miRNA sequence and its secondary structure. Some of these features were taken from our previous studies on ncRNA prediction (Wang and Wu 2006, Griesmer, Cervantes-Cervantes et al. 2011) while others were suggested in the literature (Sewer, Paul et al. 2005, Xue, Li et al. 2005, Zheng, Hsu et al. 2006). These features included the sequence length, the number of base pairs, GC content, the number of nucleotides contained in the hairpin loop (i.e., the loop size), the free energy of the sequence's secondary structure obtained from RNAfold (Hofacker 2003), the number of bulge loops, and the size of the largest bulge loop in the secondary structure.

In addition, we considered the features described in (Zheng, Hsu et al. 2006). These features included the difference of the lengths of the two tails in the secondary structure where a tail represented the strand of unpaired bases in the 5' or 3' end of the structure, the number of tails, and the length of the larger tail. Besides, several combined features were considered. They included the ratio between the number of base pairs and the sequence length, the length difference of two tails plus the larger tail length, the size of the hairpin loop plus the larger tail length, the size of the hairpin loop plus the largest bulge size, the ratio between the larger tail length and the sequence length, the ratio between the size of the hairpin loop and the sequence length, the ratio between the largest bulge size and the sequence length, the ratio between the largest bulge size and the number of base pairs, the normalized free energy (Spirollari, Wang et al. 2009), which is the minimum free energy of the sequence's secondary structure divided by the sequence length, and the ratio between the normalized free energy and the GC content.

The next set of features included the triplets of structure-sequence elements described in (Xue, Li et al. 2005). Here we used the dot-bracket notation (Hofacker 2003) to represent an RNA secondary structure. Figure 2.1 shows the sequence and structure of a hypothetical pre-miRNA and its dot-bracket notation. A triplet is composed of three contiguous structure elements (bases or base pairs) (Liu, Wang et al. 2005) that correspond to three contiguous nucleotides along with the middle nucleotide. For example, consider the first three dots (bases) and their corresponding nucleotides AAA in Figure 2.1. The middle nucleotide is A. Thus, the structure-sequence elements "A..." constitute a triplet. As another example, consider the first three brackets (base pairs) and their corresponding nucleotides UUG in Figure 2.1. The middle nucleotide is U. Thus, the structure-sequence elements "U(((" constitute a triplet. There are 32 triplets, and hence, 32 such features in total.

Finally, we considered the symmetric and asymmetric loops defined in (Sewer, Paul et al. 2005). We refer to the portion of the sequence from the 5' end to the hairpin loop as the left arm, and the portion of the sequence from the hairpin loop to the 3' end as the right arm. In a symmetric (internal) loop, the number of nucleotides in the left arm equals the number of nucleotides in the right arm. In an asymmetric (internal) loop, the number of nucleotides in the left arm is different from the number of nucleotides in the right arm. Features related to these loops included the size of each loop, the average size of the loops, and the average distance between the loops. Other features included the proportion of A/C/G/U in the stem, and the proportion of A-U/C-G/G-U base pairs in the stem. Totally, there are 74 features in the feature pool.

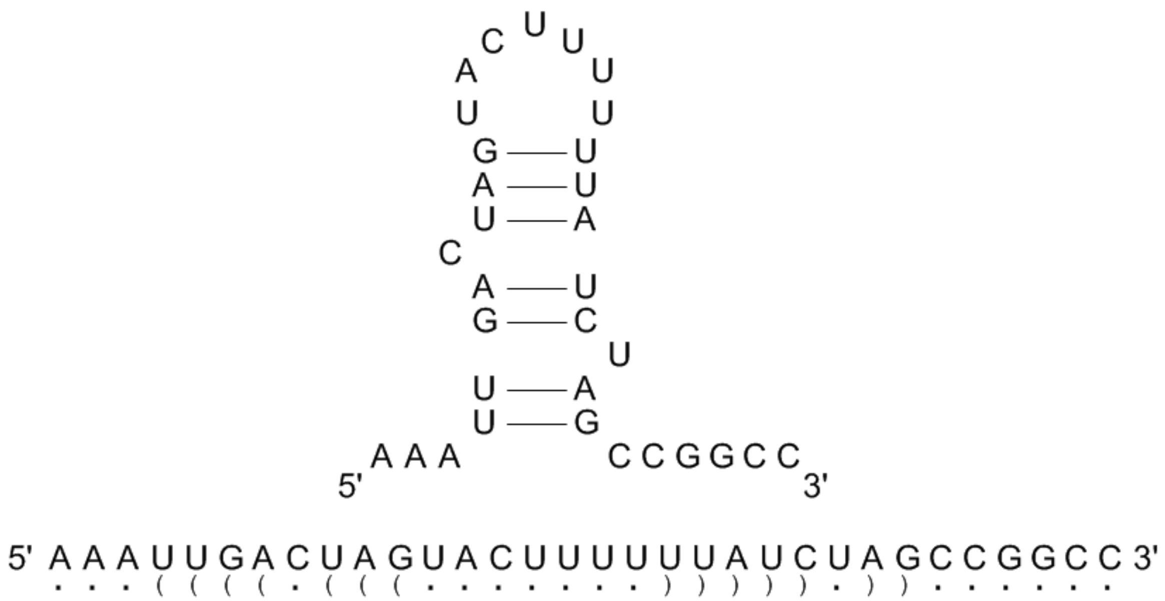


Figure 2.1 Sequence and structure of a hypothetical pre-miRNA and its dot-bracket notation.

2.2.3 Combinatorial Feature Mining

MirID adopts a combinatorial feature mining algorithm for pre-miRNA classification. Initially the algorithm randomly generates *N* feature sets from the feature pool. (The default value of *N* used in this study is 100.) Each feature set contains between 1 and 150 features, randomly chosen with replacement from the feature pool. Some features may repeatedly occur in a feature set; thus a bagging approach is used here (Breiman 1996).

Duplicate features have more weights than the other features in the feature set. The numbers 1 and 150 are chosen, to ensure that there are enough feature sets containing duplicate features. We then build a SVM model based on each feature set using training sequences, and apply the classification model to test sequences to calculate the accuracy of the model. The SVM used in this study is the LIBSVM package downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvm/ (Fan, Chen et al. 2005). We use the polynomial kernel provided in the LIBSVM package. The polynomial kernel achieves the best performance among all kernel functions included in the package.

Then, we remove the SVM models whose accuracies are less than a user-determined threshold t. (The default value of t used in this study is 0.8.) The feature sets used to build those removed SVM models are also eliminated from further consideration. We construct a classifier ensemble from the remaining SVM models. The ensemble works by taking the majority vote from the individual SVM models used to build the classifier ensemble. This ensemble will be refined through several iterations until its accuracy cannot be enhanced further. In each iteration, the user-determined threshold t is incremented by a *step* value, so that more accurate SVM models are used to construct a (hopefully) better classifier ensemble in the next iteration. (The default value of *step* used in this study is 0.005.)

It is likely that different combinations of remaining features may yield an even better classifier. Our algorithm then performs pairwise *merge* and *split* operations on the set S_b of feature sets used to build the best classifier ensemble obtained so far. In doing so, MirID takes four steps: (1) picks each pair of feature sets s_1 and s_2 in S_b; (2) merges s_1 and s_2 into a single feature set s_3 with, say p, features; (3) randomly generates a number q,

$q < p$; (4) randomly assigns q features in s_3 into a set s'_1 and assigns the remaining $p - q$ features into another set s'_2. Thus, these four steps take two feature sets s_1 and s_2 in S_b as input and produce two new feature sets s'_1 and s'_2 as output. Figure 2.2 illustrates how the merge and split operations work on two feature sets.

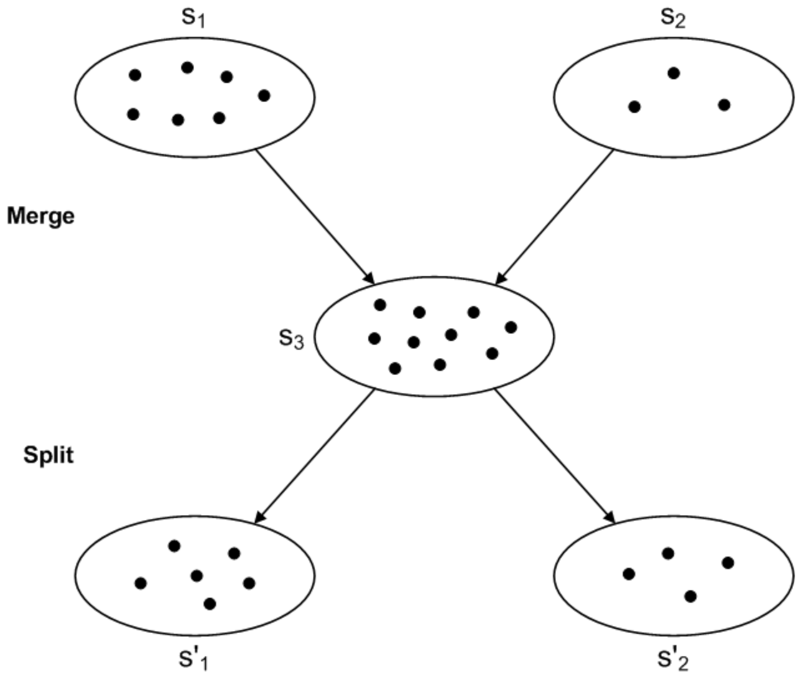
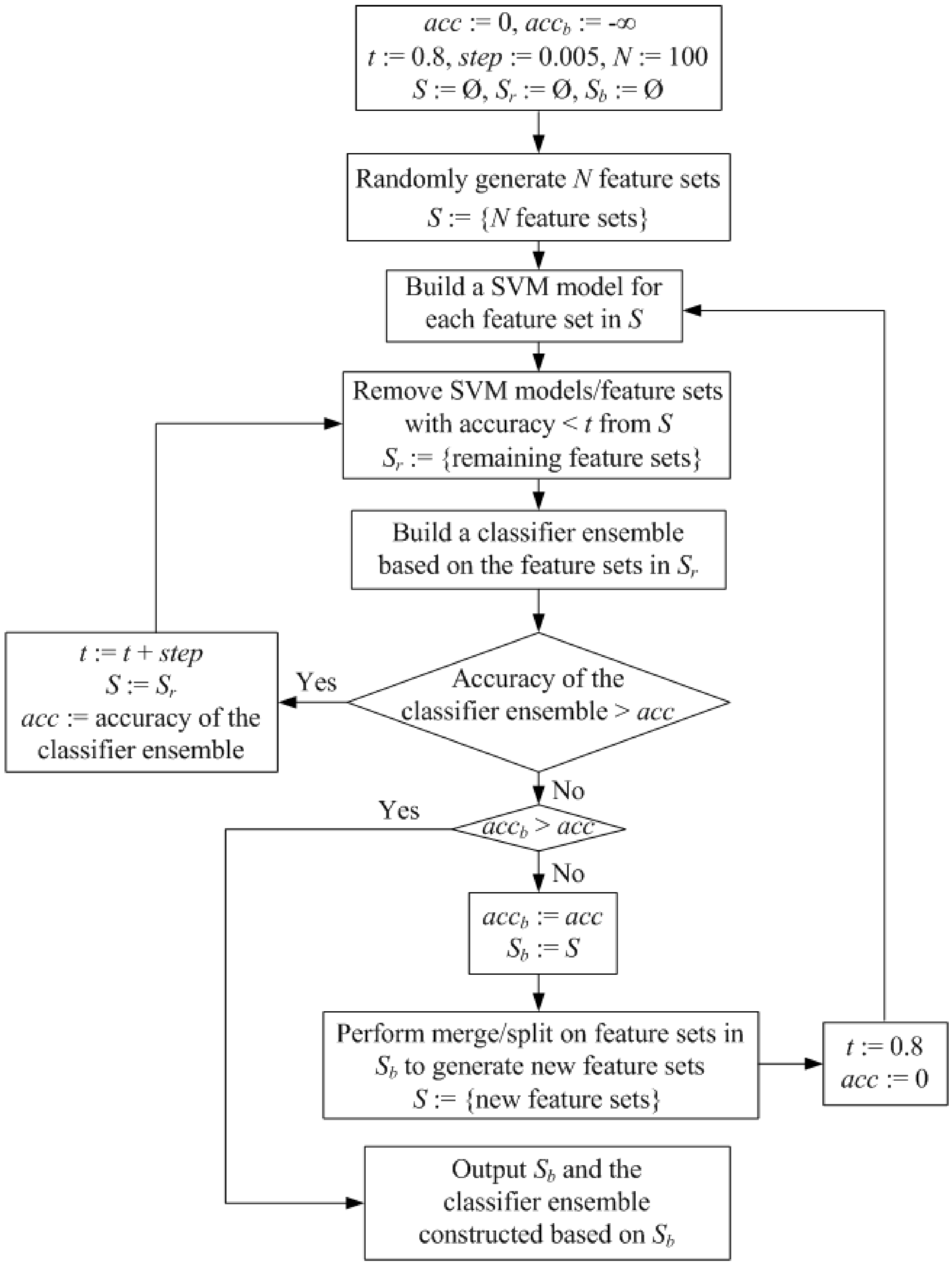


Figure 2.2 Illustration of the merge and split operations on two feature sets.

These pairwise merge and split operations are applied to the feature sets used to build the best classifier ensemble obtained so far, to generate new feature sets. The new feature sets are then used to build new SVM models. Accurate new SVM models, whose accuracies are greater than or equal to the newly computed threshold t, are then used to build a new classifier ensemble. This procedure is repeated several times to obtain a best classifier ensemble. Figure 2.3 summarizes our feature mining algorithm, whose output is the best classifier ensemble along with the component SVM models (feature sets) used to build the ensemble. Notice that in the feature mining algorithm in Figure 2.3, it is possible that, after removing SVM models/feature sets with accuracy < t, there is no remaining feature set, and hence, S_r becomes empty. Under this circumstance, the classifier ensemble constructed based on S_r is empty, and the accuracy of the classifier ensemble is 0.



$acc := 0$, $acc_b := -\infty$
$t := 0.8$, $step := 0.005$, $N := 100$
$S := \emptyset$, $S_r := \emptyset$, $S_b := \emptyset$
Randomly generate N feature sets
$S := \{N \text{ feature sets}\}$
Build a SVM model for each feature set in S
Remove SVM models/feature sets with accuracy $< t$ from S
$S_r := \{\text{remaining feature sets}\}$
Build a classifier ensemble based on the feature sets in S_r
Accuracy of the classifier ensemble $> acc$
Yes
$t := t + step$
$S := S_r$
$acc :=$ accuracy of the classifier ensemble
No
$acc_b > acc$
Yes
No
$acc_b := acc$
$S_b := S$
Perform merge/split on feature sets in S_b to generate new feature sets
$S := \{\text{new feature sets}\}$
$t := 0.8$
$acc := 0$
Output S_b and the classifier ensemble constructed based on S_b

Figure 2.3 Algorithm for combinatorial feature mining.

2.2.4 Boosting

The performance of a classification algorithm can be further enhanced through boosting. We apply AdaBoost (Freund and Schapire 1997, Schapire 1999, Bindewald and Shapiro 2006) to the classifier ensemble produced by our feature mining algorithm. Specifically, we treat the classifier ensemble as a weak classifier and continue refining it into a strong classifier through an iterative procedure. Let X be a set of sequences $x_1, x_2, \ldots, x_m$ where x_i, $1 \le i \le m$, is associated with a label y_i such that

$$y_i = \begin{cases} +1 & \text{if } x_i \text{ is a real pre-miRNA} \\ -1 & \text{if } x_i \text{ is a pseudo pre-miRNA} \end{cases} \tag{2.1}$$

The AdaBoost algorithm works with K iterations. (The default value of K used in this study is 20.) In iteration k, $1 \le k \le K$, the algorithm updates a weight function W_k as explained below, which will be used in selecting training sequences in iteration $k + 1$. Initially, every sequence has an equal weight, i.e., $W_0(x_i) = 1/m$, $1 \le i \le m$. In iteration k, the algorithm samples 1/3 sequences with replacement from X based on the weight function W_{k-1} to form a training set X_k. The set X_k is then used to train a weak classifier H_k, which classifies each sequence x_i as either a real pre-miRNA or a pseudo pre-miRNA. That is,

$$H_k(x_i) = \begin{cases} +1 & H_k \text{ classifies } x_i \text{ as a real pre-miRNA} \\ -1 & H_k \text{ classifies } x_i \text{ as a pseudo pre-miRNA} \end{cases} \tag{2.2}$$

Let E_k = [$x_i \mid H_k(x_i) \neq y_i$]. The error rate ε_k of H_k is:

$$\varepsilon_k = \sum_{x_i \in E_k} W_{k-1}(x_i) \tag{2.3}$$

Let

$$\alpha_k = \frac{1}{2} ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right) \tag{2.4}$$

The algorithm updates W_k for each sequence x_i, $1 \leq i \leq m$, as follows:

$$W_k(x_i) = \begin{cases} \frac{W_{k-1}(x_i)}{Z_k} \times e^{-\alpha_k} & \text{if } H_k(x_i) = y_i \\ \frac{W_{k-1}(x_i)}{Z_k} \times e^{\alpha_k} & \text{if } H_k(x_i) \neq y_i \end{cases} \tag{2.5}$$

$$= \frac{W_{k-1}(x_i) exp(-\alpha_k y_i H_k(x_i))}{Z_k}$$

where Z_k is a normalization factor chosen such that W_k is normally distributed. Thus, the sequences causing classification errors in iteration k will have a greater probability of being selected as training sequences for constructing the weak classifier H_{k+1} in iteration k+1. Using this technique, each weak classifier should have greater accuracy than its predecessor. The final, strong classifier H combines the vote of each individual weak classifier H_k, $1 \leq k \leq K$, where the weight of each weak classifier's vote is a function of its accuracy. Specifically, for an unlabeled test sequence x, $H(x)$ is calculated as follows:

$$H(x) = sign\left(\sum_{k=1}^{K} \alpha_k H_k(x)\right) \tag{2.6}$$

The function *sign* indicates that if the sum inside the parentheses is greater than or equal to zero, then H classifies x as positive (i.e., a real pre-miRNA); otherwise H classifies x as negative (i.e., a pseudo pre-miRNA).

2.3 Results

2.3.1 Performance Analysis of the MirID Method

We carried out a series of experiments to evaluate the proposed MirID method. All the experiments were performed on a 2 GHz Pentium 4 PC having a memory of 2G bytes. The operating system was Cygwin on Windows XP and the algorithms were implemented in Perl. To understand the effect of boosting, we also considered using the combinatorial feature mining algorithm alone to classify pre-miRNAs, and referred to it as the CFM method. The performance measure used here is accuracy, defined as follows. A method is said to classify a test sequence correctly if the sequence is a real pre-miRNA (pseudo pre-miRNA, respectively) and the method indicates that the sequence is indeed a real pre-miRNA (pseudo pre-miRNA, respectively). A method is said to classify a test sequence incorrectly if the sequence is a real pre-miRNA (pseudo pre-miRNA, respectively) but the method mistakenly indicates that the sequence is a pseudo pre-miRNA (real pre-miRNA, respectively). For each species, the accuracy of a method is defined as the number of correctly classified test sequences of that species divided by the total number of test sequences of that species.

We first evaluated how the number of initial feature sets, *N*, affects the performance of CFM and MirID. As *N* increases, more feature sets are generated initially. This allows the feature mining algorithm to construct a classifier ensemble using more diverse feature sets, and hence, the accuracy of the classifier ensemble increases. On the other hand, as *N* increases, the inner loop in Figure 2.3 is run more times; as a consequence, the running time increases. MirID requires more time than CFM, due to the extra time spent in boosting. MirID in general is more accurate than CFM, indicating the benefit of including the boosting algorithm.

We next evaluated how the threshold, *t*, used in the feature mining algorithm affects the performance of CFM and MirID. When *t* is very large (e.g., $t > 0.95$), the accuracies of the methods drop sharply. This happens because the accuracies of most SVM models are less than 0.95 (i.e., 95%), and hence, these SVM models are eliminated from further consideration early in the feature mining algorithm, cf. Figure 2.3. When *t* approaches 1, it is likely that the set S_b returned by the feature mining algorithm is an empty set, and therefore the classifier ensemble constructed based on S_b is also empty, yielding an accuracy of 0. As *t* increases, fewer feature sets qualify and the set S_r is smaller. As a result, the inner loop in Figure 2.3 is executed fewer times, and hence, the running time decreases.

Then we evaluated how the value, *step*, used to increment the threshold *t* in each iteration of the inner loop in Figure 2.3 affects the performance of CFM and MirID. With the default values of *N* and *t* used in this study, the feature mining algorithm is able to produce a classifier ensemble with high accuracy. The value of *step* has little impact on the accuracies of the proposed methods. However, as *step* increases, fewer iterations of the inner loop in Figure 2.3 are executed, and as a consequence, the running time decreases.

We also conducted experiments to test different numbers of iterations, K, in the boosting algorithm. It was found that when K is sufficiently large (e.g., $K \geq 20$), the behavior of the boosting algorithm becomes stable, with the accuracy approaching 1. On the other hand, when K is large, more running time will be needed.

Finally, we compared CFM and MirID with two closely related methods, PMirP (Zhao, Wang et al. 2010) and TripletSVM (Xue, Li et al. 2005). Like our methods, both PMirP and TripletSVM were implemented using support vector machines. PMirP adopted a hybrid coding scheme, combining features such as free bases, base pairs, minimum free energy of secondary structure, among others. TripletSVM used triplets of structure-sequence elements, which also were included in our feature pool. Table 2.2 shows the accuracies of the four methods on twelve species taken from Table 2.1. These twelve species were used to pre-train PMirP and TripletSVM, and available from the tools. For each species, the highest accuracy yielded by a tool is in boldface. It can be seen from Table 2.2 that MirID is better than or as good as the existing tools on all the species except *Gallus gallus* and *Oryza sativa.* For *Gallus gallus* and *Oryza sativa*, PMirP achieves higher accuracies.

Table 2.2 Accuracies of TripletSVM, PMirP, CFM and MirID on Twelve Species

Species	**TripletSVM**	**PMirP**	**CFM**	**MirID**
Arabidopsis thaliana	92	96	99	**100**
Caenorhabditis briggsae	96	97	98	**100**
Caenorhabditis elegans	86	86	97	**98**
Danio rerio	67	83	98	**99**
Drosophila melanogaster	92	96	97	**99**
Drosophila pseudoobscura	90	92	98	**100**
Epstein barr virus	**100**	80	98	**100**
Gallus gallus	85	**100**	96	96
Homo sapiens	93	**95**	93	**95**
Mus musculus	94	94	95	**97**
Oryza sativa	95	**100**	97	99
Rattus norvegicus	80	92	97	**98**

The unit of each number in the table is percentage (%).

2.3.2 Web Server

We have implemented MirID using Perl into a web server, accessible at https://web.njit.edu/~lz25/cgi-bin/boost/. The web server accepts a test sequence as input and classifies the test sequence as a pre-miRNA or not. We pre-train our web server using the training sequences given in Table 2.1. In addition to the twelve species available from the PMirP and TripletSVM web servers (Xue, Li et al. 2005, Zhao, Wang et al. 2010), we pre-train our web server using nine additional species (shown in Table 2.1 but not in Table 2.2). Our tool achieves high accuracies on these nine species, as shown in Table 2.3. (The PMirP and TripletSVM web servers were not pre-trained on these nine species, and hence, we only show the results for CFM and MirID here.) MirID is more accurate than CFM, due to the boosting algorithm.

Table 2.3 Accuracies of CFM and MirID on Nine Additional Species

Species	**CFM**	**MirID**
Canis familiaris	97	**100**
Ciona intestinalis	94	**100**
Macaca mulatta	**96**	**96**
Medicago truncatula	95	**100**
Physcomitrella patens	**100**	**100**
Populus trichocarpa	97	**99**
Pristionchus pacificus	96	**100**
Schmidtea mediterranea	95	**99**
Taeniopygia guttata	95	**99**

The unit of each number in the table is percentage (%).

Table 2.4 shows, for each species in Table 2.1, the number of feature sets produced by our feature mining algorithm. Table 2.5 shows the CPU time (in seconds) spent in pre-training the MirID web server. The training time depends on the number of feature sets, the number of features in each feature set, the number of iterations used by the feature mining algorithm, and the number of iterations used in the boosting algorithm. Notice that this training is done once, and no more training is needed on the test data. It takes less than a second to classify an unlabeled test sequence.

Table 2.4 Number of Feature Sets for Each Species in MirID

Species	**Number of feature sets**
Arabidopsis thaliana	1
Caenorhabditis briggsae	6
Caenorhabditis elegans	1
Canis familiaris	1
Ciona intestinalis	7
Danio rerio	11
Drosophila melanogaster	3
Drosophila pseudoobscura	4
Epstein barr virus	5
Gallus gallus	3
Homo sapiens	1
Macaca mulatta	1
Medicago truncatula	1
Mus musculus	3
Oryza sativa	3
Physcomitrella patens	1
Populus trichocarpa	1
Pristionchus pacificus	1
Rattus norvegicus	10
Schmidtea mediterranea	32
Taeniopygia guttata	5

Table 2.5 Training Time in CPU for Each Species (in Seconds)

Species	Training time (in seconds)
Arabidopsis thaliana	80
Caenorhabditis briggsae	348
Caenorhabditis elegans	103
Canis familiaris	153
Ciona intestinalis	269
Danio rerio	1272
Drosophila melanogaster	199
Drosophila pseudoobscura	196
Epstein barr virus	113
Gallus gallus	274
Homo sapiens	1530
Macaca mulatta	243
Medicago truncatula	104
Mus musculus	786
Oryza sativa	214
Physcomitrella patens	90
Populus trichocarpa	138
Pristionchus pacificus	63
Rattus norvegicus	349
Schmidtea mediterranea	478
Taeniopygia guttata	156

2.4 Discussion

In this chapter, we present a new method (MirID) and a web server for pre-miRNA classification. Empirical results showed that MirID outperforms two closely related methods, PMirP and TripletSVM, on the majority of species tested in the experiments. Since all the three methods were implemented using support vector machines with similar features, we conclude that the superiority of our method is due to its novel feature mining and boosting algorithms.

Both the feature mining and boosting algorithms contain user-specified parameters. As indicated by our experimental results in the performance analysis section, changing these parameter values may affect the running time and accuracy of our method.

The MirID web server adopts the default parameter values as used in this study, to achieve good and stable performance. The server is able to process sequences of a variety of species, from viruses to humans. It does not include bacteria, however. While there are small regulatory RNAs in bacteria, bacteria do not have true miRNAs (Gottesman 2005, Tjaden, Goodwin et al. 2006). Bacterial miRNA will be added to our server when such data is validated and becomes available in public databases.

Currently, the MirID web server is capable of classifying one test sequence at a time, predicting whether the test sequence is a pre-miRNA or not. When multiple test sequences must be classified, we suggest that the user run the tool locally in a batch mode. Instructions for downloading the tool and running the tool locally can be obtained from https://web.njit.edu/~lz25/cgi-bin/boost/MirID-download.

MicroRNAs play important roles in most biological processes, including cell proliferation, tissue differentiation, embryonic development, to name a few (Aukerman (Aukerman and Sakai 2003, Brennecke, Hipfner et al. 2003, Johnston and Hobert 2003, Bushati and Cohen 2007, Tang, Zhang et al. 2009, Xu, Yu et al. 2009). They interact with target mRNAs at specific sites to induce cleavage of the message or inhibit translation (John, Enright et al. 2004). They can have multiple mRNA targets as they bind to the targets with partial complementarities in animals. In addition, an mRNA target can be regulated by multiple miRNAs at different loci with different effects. This adds to the complexity of finding out the mRNA targets in genomes (John, Enright et al. 2004).

The total number of microRNA discovered continues growing every day. According to the latest miRBase release (version 19, August 2012), accessible at http://www.mirbase.org, there are 2,019 unique mature human miRNAs up from 894 in the version 14. There seems to be a correlation between the tissue-specificity of a human miRNA and the number of diseases the miRNA is associated with (Lu, Zhang et al. 2008). The fact that microRNAs are found circulating in blood (Mitchell, Parkin et al. 2008, Scholer, Langer et al. 2011) holds great promise for the development of diagnostic tools that can be used in multiple ways, from non-invasive pregnancy diagnostic tests to cancer diagnostics and treatment. A tool like MirID for predicting pre-miRNAs will contribute to our basic understanding of the roles played by microRNA in regulating many biological processes, and their contribution to disease development and progression.

A potential application for the MirID tool is in the area of individualized genomic analysis. With the advent of high-throughput sequencing technologies, millions of short reads can now be generated from a library of nucleotide sequences. These technologies have catalyzed a new era of personalized medicine based on individualized genomic analysis (Anderson and Schrijver 2010) Determining levels of known and novel microRNA from small RNA sequencing data is an important subject in this new era (An, Lai et al. 2013). With next-generation sequencing platforms, several prostate expressed microRNAs related to prostate cancer have been identified (Ribas, Ni et al. 2009, Ostling, Leivonen et al. 2011, Wang, Chatterjee et al. 2011, Watahiki, Wang et al. 2011, Martens-Uzunova, Jalava et al. 2012). As a consequence, exploring microRNAs and their functions continues to be a highly active area of research. The MirID tool developed from this work can be used to assess aggregated RNAseq reads for pre-miRNA secondary structure potential. The tool can be combined and integrated with other miRNA profiling tools (Hendrix, Levine et al. 2010, Mathelier and Carbone 2010, Hackenberg, Rodriguez-

Ezpeleta et al. 2011, Friedlander, Mackowiak et al. 2012) for applications to personalized medicine.

CHAPTER 3

ALGORITHM FOR GENE NETWORK INFERENCE: A SURVEY

3.1 Background

In living organisms, cells contain thousands of genes, working in concert to direct the cells' functions while ensuring their fitness, multiplication, and survival. Whereas some genes are continuously expressed, others only do so in response to specific stimuli, at the right time, and to the proper extent, thus ensuring appropriate functional outcomes. Some genes have highly robust regulation mechanisms of their expression, which is controlled by stringent programs. In eukaryotic species, for example, the control of developmental gene expression is significantly similar in a given cell type from one individual to another (Macneil and Walhout 2011). Nonetheless, the timing and scale for the expression of other genes can be more variable, resulting in expression levels that frequently change and which differ from cell to cell and from individual to individual. Research on gene expression directing physiological responses to developmental cues and environmental stresses is, therefore, greatly beneficial. Currently, we are focusing on the analysis of differential gene expression at the level of systems biology. Gene regulatory networks (GRNs) illustrate interactions between large numbers of genes and their regulatory mechanisms. Graphic diagrams are applied to map all the interactions and visualize the regulatory relationships. Further characterization of GRNs has already uncovered global principles of gene regulation (MacNeil and Walhout 2011).

Specifically, we are using computing algorithms to infer gene regulatory networks according to the expression values of genes and their changing platforms. This can be accomplished with basis on inferring causality. If gene A can cause gene B to switch to a high expression value, then B can be stopped from taking such a value after making B less responsive or A less active, or by interfering with the link from A to B. Conversely, B can achieve a higher expression value if A’s expression value is increased itself, or by enhancing the efficiency of the link from A to B. We call such a relationship a causal link. The expression of gene A may influence the expression of gene B as follows: gene A is transcribed to RNA and then translated to a protein, which in turn may bind to the promoter of B, either allowing or preventing the transcription of gene B. Comparing the expression values of A and B under particular circumstances versus wild-type data (i.e., expression values from the most common phenotype of a given organism), it can be determined whether or not there is a relationship between genes A and B. Based on such predictions, a gene regulatory network can be graphed (Lingeman and Shasha 2012).

In this chapter, we review state-of-the-art algorithms for the inference of gene regulatory networks (GRNs) from microarray gene expression data. A gene regulatory network is represented by a directed graph, in which nodes represent transcription factors or genes and an edge represents the transcriptional regulation relationship between two genes. The algorithms for GRN inference can be categorized into four groups: unsupervised, semi-supervised, supervised and integrated methods. In unsupervised algorithms, a network is unknown, and the algorithms predict the entire network using time series or steady-state gene expression data. For supervised and semi-supervised algorithms, a portion of a network is known in advance, possibly from publicly available databases, and the algorithms use that portion as prior knowledge to predict remaining edges in the network. Integrated methods combine unsupervised or supervised algorithms, coupled with prior knowledge from literature mining or information integration techniques. Here, we survey the various techniques employed in the unsupervised, semi-supervised, supervised and integrated methods, and present a taxonomy of existing algorithms.

3.2 Unsupervised GRN Inference Algorithms Based on Steady-State Data

In this section we review six algorithms for inferring gene regulatory networks using steady-state data. If the expression values of genes of one organism will not change unless its conditions are changed in some way, this organism is said to be in steady state. For instance, an organism is in a certain steady state if it under a low nutrients condition; another state under a high nutrients condition; and yet another, if some mutation has happened or transient effects have changed or disappeared altogether. Steady-state data can be obtained from experiments where one or more genes have been knocked out or from reported expression values that have been significantly changed or perturbed in other way. If some changes in the network can be noticed when one gene is absent or it has been perturbed, one can determine which other gene or genes it influences.

Many published algorithms were tested using the data are from DREAM4 in silico datasets (Greenfield, Madar et al. 2010). DREAM stands for Dialogue for Reverse Engineering Assessments and Methods and provides a set of networks that can be used to develop and test GRNs. The networks presented by DREAM make some simplifications of naturally occurring networks found in a cell, and the corresponding datasets are ideal in their completeness. The datasets include results from knockout and knockdown experiments, multifactorial perturbations, time series and dual knockouts. These datasets are considered as standard benchmark data in the field.

3.2.1 Network Identification by Multiple Regressions (NIR)

The way the Network Identification by Multiple Regression (NIR) (Gardner, di Bernardo et al. 2003) infers gene networks is by using multiple regression. It uses steady-state data resulting from a known initial perturbation. Basically, we assume that a gene network can be described with a series of linear equations, approximately:

$$dX/dt = AX + U \tag{3.1}$$

where X is an n by m matrix of steady-state expression data. In X, each column represents an experiment and the rows represent genes. A is an n by n normally distributed matrix that represents the network model, which implies that every gene's expression is a linear function of the sum of a row of coefficients from A and the gene values as a column from X. U is an n by m matrix which represents the degree to which the gene is perturbed in each experiment (values from 0 and 1). For example, the degree would be 1 if the gene is totally knocked out. If a gene is knocked down, it might have a value of 0.5. Genes have values of 0 if they are not perturbed at all. dX/dt shows how the expression values change per unit of time. As NIR is applied with steady-state data (which means data would change little time by time), dX/dt is 0. Therefore, the above equation can be reduced to:

$$-U = AX \tag{3.2}$$

Our goal is to select a promising network model A by using multiple regressions. From the beginning we just take one row a_i from A and one column x_j from X, and try to solve u_{ij} of U. We need to achieve all the values in a_i, and multiplied by x_j, to have the sum of all results equals to –u_{ij}. We will aim to get the best solution, as there would be many possible answers.

A multiple linear regression model that can account for more than one independent variable will be created as a solution in NIR. The independent variables consist of one possible set of k out of n genes, where k is a user-defined parameter that enforces sparsity in A in order to limit the number of dependencies between genes, and –u_{ij} is the dependent value for the target gene/experiment. We repeat all these steps for each gene/experiment combination. The solution matrix A is derived from the model in which each gene has best weight.

NIT uses least squares regression to minimize the sum of squared errors (SSE):

$$SSE_i^k = \sum_{l=1}^{m} (y_{il} - b_i^T \cdot z_l)^2 \tag{3.3}$$

where k represents the number of genes being examined, i is the target gene, l is the current experiment, y_{il} is the negative perturbation value for gene i in experiments l, b are the model weights for gene i, and z_l are the expression values from the currently selected k genes in experiment l. Our goal is to choose weights b to minimize the sum of squared errors. In fact, the squared error presents the difference between how much the target gene was perturbed and the perturbation that the current model shows. For example, if the gene under a current perturbation has U have a value of 1, then we propose to find a number k of weight b (at least one is non-zero) whose dot product with the current expression values is equal to –1 (since y_{il} = –u_{il}), which makes the error 0. All the edges with a non-zero weight indicate that the source genes regulate the target gene i. The basic technique to choose b weights consists of randomly initiating their values, then refining them.

3.2.2 GEne Network Inference with Ensemble of Trees (GENIE3)

GENIE3 (Huynh-Thu, Irrthum et al. 2010) is an algorithm by using an ensemble of regression trees to predict networks. All the expression data applied to the GENIE3 algorithm are normalized to unit variance (i.e., variance of 1) and mean 0. There are three steps in the algorithm: 1) builds up an ensemble of regression trees for each gene; 2) ranks potential regulator in each regression tree; 3) ranks all of the inferred edges.

At first, GENIE3 creates a regression tree based on the mutation of expression values of each gene *g* in the network. To create regression trees, the whole dataset is split recursively into smaller subsets. According to the expression values of genes other than *g*, the dataset will be split on the nodes of the regression tree. To avoid confusion with the term node in the final result, the regression tree nodes here are called decision points. Each sub-dataset divided by a decision point has a small variance in the target gene's expression values. For a single gene *x* other than *g*, we pick up a threshold, based on which the division is made, according to the idea that if gene *x* causes a split in the regression tree for target *g*, there is a potential causality from *x* to *g*. Then the expression value of gene *x* in each experiment will be checked. The experiment with the expression value of *x* above the threshold goes to one group or to another group if the expression value is below the threshold. The splitting continued recursively until no more splits can be made.

As an example, consider Table 3.1, which shows an example of data for GENIE3. The basic idea of the GENIE3 algorithm is to split all the experiments into two groups according to expression values so that each group has minimal variance on the target gene.

Table 3.1 Sample Data for GENIE3

Experiments	Genes			
	G1	**G2**	**G3**	**Target**
E1	0.4	0.8	0.4	0.5
E2	0.3	0.2	0.3	0.9
E3	0.5	0.3	0.7	0.8

Source: Lingeman, J. M. and D. Shasha (2012). *Network inference in molecular biology*. New York, Springer.

In Table 3.1, the ideal split is to have experiments E1 alone in one group, and E2 and E3 in the other group. It is clear that G2 is the potential source gene because G2's value in E1 is distinguished from its value in E2 and E3. Here the threshold is defined as 0.5 as it is between G2's E1 and E3 values. Any values above 0.5 go to group 1, and any values less than or equal to 0.5 go to group 2.

Table 3.2 Example Data for GENIE3

Experiments	Genes			
	G1	**G2**	**G3**	**Target**
Group 1	0.4	0.8	0.4	0.5
Group 2	0.3	0.2	0.3	0.9
	0.5	0.3	0.7	0.8

Source: Lingeman, J. M. and D. Shasha (2012). *Network inference in molecular biology*. New York, Springer

Table 3.2 shows example data for GENIE3 with the dataset split into two groups, thus minimizing the variance of each of the groups.

When the expression value of G2 is above 0.5, the target gene has low expression values, whereas with a value below 0.5, the target gene displays high expression values. Thus, we have identified a potential causal edge, in which G2 has a repressive effect on the target gene.

The algorithm of Random Forests (Breiman 2001) is applied here to find robust splits in their responses to trivial changes in the data. With averaging predictions, Bootstrapping and random feature selection are used in Random Forests and reduce variance across the dataset. From the original dataset, 2/3 data is randomly picked up with replacement to generate every tree in a Random Forest ad a bootstrap training. Each tree is built later with K random splits at each decision node. Here K is normally defined as $K = \sqrt{p-1}$ or $K = p$ -1 where p is the number of known potential regulators also termed as transcription factors. The randomly chosen split which will reduces mostly the variance of the target gene's expression values is defined as a decision split.

We calculate an importance score for each decision point in a tree:

$$I(N) = \#S Var(S) - \#S_t Var(S_t) - \#S_f Var(S_f) \tag{3.4}$$

where N is the current decision point to be evaluated, S is the subset of experiments lower in the tree than the decision point N, S_t is the subset of experiments on the true branches of decision point N, and S_f is the subset of experiments on the false branches. Var(.) denotes the variance of the target gene in a subset, and # is defined as the number of experiments in corresponding subset. The importance score is to measure how much variance shown by splitting the dataset on the gene at the decision point with threshold. If the score is high, it presents that the variance is significantly reduced and probably this gene regulate the target gene, as shown in the example above. If the score is low, the split did not considerably reduce the variance which means the gene may not regulate the target gene.

After the tree for gene *g* is generated, the influence of every other gene on *g* is ranked. By summing up all of the importance scores of the nodes at which a potential regulator gene *x* was selected to split, we can get a final score for *x*. A zero score is appointed to the genes that are never selected to split. All the final scores of potential regulator genes are ranked to define which genes *x* are most crucial to regulate gene *g*.

3.2.3 Relevance Networks

The tool of relevance networks was developed by Butte and Kohane (Butte and Kohane 2000). It measures the mutual information (*MI*) between gene expression profiles to infer interactions. Let X_i represent the vector of expression values of gene *i* and let X_j represent the vector of expression values of gene *j*. The mutual information *I* between the discrete variables X_i and X_j is defined as:

$$I(X_i, X_j) = \sum_{x_i \in X_i} \sum_{x_j \in X_j} p(x_i, x_j) \log\left(\frac{p(x_i, x_j)}{p(x_i)p(x_j)}\right) \tag{3.5}$$

where p$(x_{i,}y_j)$ is the joint probability density function between X_i (the expression profile of gene *i*) and X_j (the expression profile of gene *j*), and $p(x_i)$ and $p(y_j)$ are the marginal probability density functions of X_i and X_j, respectively. Marginal probability density functions collect the probability densities of a subset of the data, which denotes the expression profiles of each gene, and the functions are used to present how likely *x* the expression value is in the expression profile of its gene *X*. In a expression profile *X*, if a probability of *x* is low, it would be expected by other values in *X*. Whereas, if *x* is high, it means there is a potential edge between genes X and Y. Here X_i and X_j are required to be discrete variables. The approach of equal-width binning (Meyer, Lafitte et al. 2008) is used for discretization and empirical entropy estimation.

3.2.4 Context Likelihood of Relatedness (CLR)

The basic idea of Context Likelihood of Relatedness (CLR) (Faith, Hayete et al. 2007) is to form a matrix of mutual information scores by calculating the mutual information between each pair of genes in the network. These scores are then compared to a background distribution and a z-score is calculated. A potential mutual information edge will be indicated between the two genes in the network with a high z-score, as a low z-score shows up no edge existing. Only undirected edges can generated by CLR because of the bidirectional nature of mutual information. If we have discovered the transcription factors, the directionality can be estimated based on one of the genes is exactly a transcription factor.

At the beginning, the mutual information can be calculated as shown in relevance networks in the section of 3.2.3. After a matrix of mutual information scores for each gene is calculated, the likelihood of each pair of scores (a z-score) is estimated by CLR, and they are compared with a background mutual information distribution, namely MI_i and MI_j, respectively. The two distributions represent only one row of the mutual information values of gene i and gene j. Our plan is to check the distance between a mutual information score and the rest of the mutual information scores other than that gene. If the score is much higher than most of the other scores, an edge is most likely here. As we mentioned before, all genes only depend on a small number of other genes, hence, the scores of all the genes can be used as a background distribution. Most of the mutual information scores can be used as a background noise distribution as they would not equal to zero because of measurement noise or indirect edges. After the z-score for each pair of genes has been calculated, the final step is to calculate the CLR score which is calculated for each pair of genes:

$$f(Z_i, Z_j) = \sqrt{Z_i^2 + Z_j^2} \tag{3.6}$$

Z_i and *Z_j* are the z-scores computed from the background distribution above. $f(Z_i, Z_j)$ is used to measure the joint likelihood, which gives out a single score for each pair of genes compared to the score of each other pair of genes. In the final step, the CLR scores are ranked and the top *N* scores are applied to generate a network. However, we must choose the value of *N* carefully, because we cannot confirm the N is invalid or not if the gold standard is unknown. Actually, with the computational analyst the ranking can be provided as the topmost ranked genes will be tested by the experimentalist.

3.2.5 EULID

EUCLID (Maetschke, Madhamshettiwar et al. 2013) is an approach by using the Euclidean distance between the normalized expression profiles X'_i and X'_j to indicate the interaction weights between gene *i* and gene *j*:

$$w_{ij} = \sqrt{\sum_k (X'_{ik} - X'_{jk})^2} \tag{3.7}$$

In this method, profiles are normalized by computing the absolute difference of expression values X_{ik} to the median expression in profile $X'_{ik} = |X_{ik} - median(X_i)|$, and *k* is from 1 to *m* (the number of experiments).

3.2.6 Weighted Gene Co-expression Network Analysis (WGCNA)

WGCNA (Langfelder and Horvath 2008) is a collection of inference methods which are correlation-based to amplify high correlation coefficients by increasing the absolute value to the power of β ('softpower') with $\beta \geq 1$:

$$w_{ij} = \left|corr(X_i, X_j)\right|^{\beta} = \left|\frac{cov(X_i, X_j)}{\sigma_X \sigma_Y}\right|^{\beta} = \frac{E\left(X_i X_j\right) - E(X_i)E(X_j)}{\sqrt{E\left({X_i}^2\right) - E^2(X_i)}\sqrt{E\left({X_j}^2\right) - E^2(X_j)}} \quad (3.8)$$

The network predicted by WGCNA is an undirected graph. We choose the node (which is a gene) with the highest amount of edges as a regulator, so that all the edges attached to it are directed to all its neighbors. Here, w_{ij} represents a weighted edge between gene i and gene j, cov is covariance, E is expected value, the sum of each expression value multiplied with its probability. With the network constructed, genes are clustered. Basically, WGCNA uses the algorithm of correlation network.

3.3 Unsupervised GRN Inference Algorithms Based on Time-Series Data

Time-series data collect the information of the values of genes at a series of time points in succession. With this temporal information, we can try to infer directionality of edges, or extract causal relations between genes. In this section, we present three algorithms for inferring gene regulatory networks from time series data.

3.3.1 Time-delay ARACNE

Time-delay ARACNE (Zoppoli, Morganella et al. 2010) is an algorithm based on mutual information to detect time series networking. There are three steps in the algorithm (Margolin, Nemenman et al. 2006). At first, the expression values of each gene are scanned and the time point at which the values suddenly vary is found. Then, a ratio of the change of expression values is used to determine whether the gene is induced or repressed. Second, the mutual information value is calculated for each gene pair. An edge is created if there is a nonzero mutual information value comes out. Third, all the edges created are tested with a certain threshold, and the edges with low data processing inequality are pruned away.

The algorithm evaluates the first time point at which a gene g is induced or repressed by comparing the expression value at time 1 to the expression value at time *t*. The parameter τ is defined as a threshold then if the expression value is varied above this amount, the gene is considered being induced or repressed.

If the ratio between the expression at *t* and the initial expression value is greater than τ, then the gene is considered expressed at time *t*, as follows:

$$g^{+}\ if\ \tau < \frac{g(t)}{g(l)} \tag{3.9}$$

On the contrary, once the ratio between the expression value at *t* and the initial expression value is lower than 1/ τ, the expression of this gene can be considered as repressed.

$$g^{-}\ if\ \frac{1}{\tau} > \frac{g(t)}{g(l)} \tag{3.10}$$

If the expression value does not meet either of these two conditions, it would not be considered expressed or repressed at time *t*.

We have three advantages to evaluating at which time point a gene start to be induced or repressed: First, it allows us to infer causal events in the time series. Gene g can effects *x* only if g is induced or repressed before gene *x*. Second, computation time can be speed up by reducing the number of possible edges to calculate in previous step. Third, dependencies between genes can be detected at each time point. This is followed by network building and trimming, using Data Processing Inequality.

3.3.2 Banjo

Banjo (Yu, Smith et al. 2004) stands for Bayesian Network Inference with Java Objects. This is a special tool because the expression values of each gene can be predicted from its parents' expression values and some values of itself at previous time points. Basically, Banjo needs to search all possible networks, and has a score system to pick up the best one. The score system includes two metrics, the Bayesian Dirichlet Equivalence (BDE) and the Bayesian Dirichlet Criterion. It also includes some search strategies which are combined with the scoring matrices: greedy, simulated annealing, and genetic algorithm. Based on previous uses on Banjo, the greedy search algorithm with Bayesian Dirichlet Equivalence (BDE) scoring will be used as defaults (Elati, Neuvial et al. 2007). BDE used here is calculated with the log of the marginal likelihood $P(D|G)$, where D is data and G is network graph.

3.3.3 Granger Causality

Mukhopadhyay (Mukhopadhyay and Chatterjee 2007) and Shojaie (Shojaie and Michailidis 2010) introduced a method called Granger Causality to analyze time-series gene expression data to predict gene networks, which is a statistical hypothesis approach to decide if one time series is useful to predict other time series. The basic idea is that a time series X is said to Granger-cause Y if it can be shown, usually through a series of t-tests and F-tests on lagged values of X (and with lagged values of Y also included), that those X values provide statistically significant information about future values of Y. Here X is the time vector of expression values of the source gene, and Y is the time vector of expression values of the target gene. To test the null hypothesis that X does not Granger-cause Y, we need to find the proper lagged values of Y to include in a formal auto regression of Y:

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_m y_{t-m} + r_t \qquad (3.11)$$

Here r_t is residual, and it is the difference between the observed expression value and the estimated function value. Next, the autoregression is augmented by including lagged values of X:

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_m y_{t-m} + b_1 x_{t-1} + \cdots + b_q x_{t-q} + r_t \qquad (3.12)$$

In this expression, y_t and x_t are stationary time series values of Y and X at time t, x_{t-j} ($0 \leq j \leq q$) is the value of X at time t-j, y_{t-i} ($0 \leq i \leq m$) is the value of Y at time t-i; a_i and b_j ($0 \leq i \leq m$, $0 \leq j \leq q$)are the coefficient values we need to find by autoregression. And q is the longest lag length when the lagged value of x is remarkable, and m is the longest lag length when the lagged value of y is remarkable. We retain in this regression all lagged values x's ($x \in X$) that are individually remarkable based on their t-statistics, given that collectively they add explanatory power to the regression according to an F-test (whose null hypothesis offers no explanatory power when jointly added by the x's, which means X and Y has no relationship.). The t-statistics for each individual a_i and b_j is a ratio of the difference of the regression result from its notional value (normal zero, indicated that no relationship) and its standard error, which can be presented as follows:

$$t_{\hat{\beta}} = \frac{\hat{\beta} - \beta_0}{s.e.(\hat{\beta})} \tag{3.13}$$

where β_0 is a known constant normally set to zero, and $\mathrm{s.e.}(\hat{\beta})$ is the standard error of the estimator $\hat{\beta}$, which is either a_i or b_j. The F-test can be calculated as:

$$F = \frac{SSR_X - SSR_Y}{SSR_Y}(T-1) \tag{3.14}$$

$\mathrm{SSR_X}$ and $\mathrm{SSR_Y}$ are the sum of squared residuals of *X* and *Y*, respectively. Residuals present the difference between estimated values and actual values. *T* is the total number of time point. In the notation of the above augmented regression.

3.3.4 DDGni

DDGni stands for Dynamic delay gene-network inference, applying for high-temporal data by using gapped local alignment. The basic idea is that, if gene *A* is regulating gene *B*, the expression pattern of the target gene *B* is stimulated by the expression pattern of its regulator gene *A*, i.e., they share similar expression pattern with some time windows (expression delay). The method also considers the cases of multi-regulators and multi-targets with different time windows.

To compute the gapped local alignment of expression patterns of gene *A* and gene *B*, we need to extract their expression values at all time pointes, consider *A* has *x* number of time points, while *B* has *y* number of time points.

$$A = a_1, a_2, a_3 \ \ldots, a_x B = \ b_1, b_2, b_3, \ldots, b_y \tag{3.15}$$

Since a regulator and its target share a similar expression trend despite the magnitude of the variations of expression values, all the expression values (*a's* and *b's*) are normalized first.

$$s'(i,j) = \frac{s(i,j) - \bar{s}(i,j)}{s_{max}(i,j)} \tag{3.16}$$

where

$$s(i,j) = e^{-\alpha \times d(a_{i,}b_j)}, 0 \leq i \leq x, 0 \leq j \leq y \tag{3.17}$$

Here α is the measure of the steepness which is set as 1.7, and $d(a_{i,}b_j)$ is the distance between time points a_i and b_j.

Then an alignments matrix *M* (*x* by *y*) is created:

$$M_{ij} = max \begin{Bmatrix} M_{i-1,j-1} + s'(i,j) \\ M_{i-1,j} - p \\ M_{i,j-1} - p \\ 0 \end{Bmatrix} \tag{3.18}$$

M_{ij} is the score for position *i* in series *A* and position *j* in series *b*, *s'(i,j)* is the normalized similarity between the time point a_i and b_j, and *p* is the gap penalty, chosen as 0.3. The alignment score *N* is calculated as:

$$N = \left[\frac{Max(M)}{L}\right] M_{ij} = max \begin{Bmatrix} M_{i-1,j-1} + s'(i,j) \\ M_{i-1,j} - p \\ M_{i,j-1} - p \\ 0 \end{Bmatrix} \tag{3.19}$$

where *L* is the alignment length. By choosing up a threshold for *N*, the top high alignment scores detemind the edge of the regulatory network.

3.4 Unsupervised GRN Inference Algorithms Using Pipelines

For gene network inference, various algorithms can be used to build up a pipeline, and each algorithm is fed with different types of data, in turn based on different experiments within the same gene network. A consensus network can be generated by combining the inferential abilities of different methods. Also, a pipeline can be a sequence of algorithms in which the output of one algorithm becomes the input of the next algorithm in the sequence. All these attempts are made to get a better performance for predicting gene regulatory networks. For example, Inferelator 2.0 (Madar, Greenfield et al. 2009) is a pipeline combining three algorithms: Median-Corrected Z-Scores (MCZ), Time-lagged Context Likelihood of Relatedness (tlCLR), and Inferelator 1.0.

3.5 Supervised or Semi-Supervised GRN Inference Algorithms

Maetschke et al. (Maetschke, Madhamshettiwar et al. 2013) defined supervised, semi-supervised and unsupervised algorithms for inference of gene regulatory networks. Usually, there is no data can be used by unsupervised methods to adjust internal parameters. Supervised methods, on the other hand, by collecting information about known interactions for training and testing all given data, both positive and negative training samples, optimize parameters such as weights or thresholds. Otherwise, only part of the data can be used by semi-supervised methods for parameter optimization, i.e., a subset of network interactions discovered, sometimes even only positive training samples. The only method described in their paper is Support Vector Machine (SVM) (Cortes and Vapnik 1995). By applying containing supervised learning models which are associated learning algorithms to analyze data and recognize patterns, this method can be used for classification and regression analysis. With the input of a set of input data, SVM predicts each one of them belonging to either of two possible classes, so it is a non-probabilistic binary linear classifier. Provided a set of training examples, each of which is labeled to one or other classes, the training algorithm of generates a model to label new examples with one category or the other. An SVM model presents the examples as points in space, and makes a gap as big as possible to divide all the examples to separate categories. Based on which side of the gap the examples to be predicted fell on, they are labeled with a category after being mapped into the same space.

Mordelet et al. (Mordelet and Vert 2008) have developed a method for the inference of gene regulatory networks called SIRENE (Supervised Inference of Regulatory Networks). It needs both gene expression data and a list of regulation relationships discovered between transcription factors and target genes. The list is obtained from the databases of regulations which are characterized with experiments and publicly available, e.g., RegulonDB for *Escherichia coli* genes (Faith, Hayete et al. 2007). To get negative samples, they collected all genes that are not regulated by a certain transcription factor and divided them into three subsets. Then they trained the SVM with all the positive examples and two of the three subsets of negative samples. The training process was repeated three times and each on took just one subset of negative samples apart.

In a paper by Cerulo et al. (Cerulo, Elkan et al. 2010), three methods are compared for predicting gene regulatory networks from only positive and unlabeled data derived from the tool *GeneNetWeaver* (http://gnw.sourceforge.net), which is used to generate in silico benchmarks in the DREAM3 challenge initiative (Stolovitzky, Monroe et al. 2007, Marbach, Schaffter et al. 2009). The three methods are considered semi-supervised methods and they are *PosOnly*, *PSEUDO-RANDOM*, and *SVMOnly*. *PosOnly* uses a model of conditional probabilities to define negative samples (Elkan and Noto 2008). Let x be a feature vector and let $y = \{0, 1\}$ and $s = \{0, 1\}$ be binary labels. Let $s = 1$ if the example x is labeled, and let $s = 0$ if x is unlabeled. Positive examples are labeled, i.e., if $s = 1$, then $y = 1$; while unlabeled examples, with $s = 0$, may be either positive $y = 1$ or negative $y = 0$. The probabilistic binary classifier is applied to learn $f(x)$ such that $f(x) = p(y = 1|x)$, defined to be the conditioned probability of being positive provided a feature vector x. *PSEUDO-RANDOM* selects negative samples according to an assumption that a regulatory network has a structure like a tree without or containing few cycles. *SVMOnly* considers all unlabeled examples as negative. The result showed that *PosOnly* outperforms significantly both methods *PSEUDO-RANDOM* and *SVMOnly* in simulated data, while the other two tools have a slightly lower performance in experimental data.

3.6 Integrated Approaches to GRN Inference

Marbach et al. (Marbach, Roy et al. 2012) have developed and applied methods for transcriptional regulatory network inference from diverse functional genomics datasets, and have demonstrated the effectiveness of their approach for gene function and gene expression prediction. The network inference problem is formed to a machine-learning framework, with input features consisting of transcription factor (TF), evolutionarily-conserved sequence motifs, gene expression, and chromatin modification datasets, to predict regulatory edges by binding all the above features. The authors predicted ~300k regulatory edges in a network of ~600 TFs and 12k target genes by applying these methods to *Drosophila melanogaster*. An inferred network is applied to identify putative functions for hundreds of previously unlabeled genes (Lee and Tzou 2009), as a lot of these genes are in nervous system development defined independently according to the patterns of tissue-specific expression. At last, the regulatory network is used as a function of TF expression to predict the levels of target gene expression, and in integrative networks remarkably better performance of prediction is achieved than for motif or ChIP-based networks. A compatible relationship is discovered by their work between physical evidence of regulatory interactions such as TF binding or motif conservation and functional evidence like coordinated expression or chromatin patterns, and the power of data integration is revealed for network inference based on the studies of gene regulation at the systems level.

Literature mining is another way to develop gene regulation networks by collecting the information about gene interactions from previously published literature. If some research already published shows that a gene can regulate other genes based on biological experiments, this scene would be applied to the basic part of the network. All pertinent information about a particular gene family can be searched, collected, and postulated as a gene regulation network.

Djebbari et al. (Djebbari and Quackenbush 2008) described in their paper how they did such a search in PubMed (McEntyre and Lipman 2001): two genes are assumed between them may be an interaction if both and only two of them are described in a single article indexed in PubMed. According to the relative number of articles talking about those genes together, weights would assigned to interactions. The prior probabilities for two genes A and B related is shown by assigning a co-occurrence edge weight, which obtained by summing up how many times the work "interaction" mentioned in the literature, relative to the total number of manuscripts surveyed:

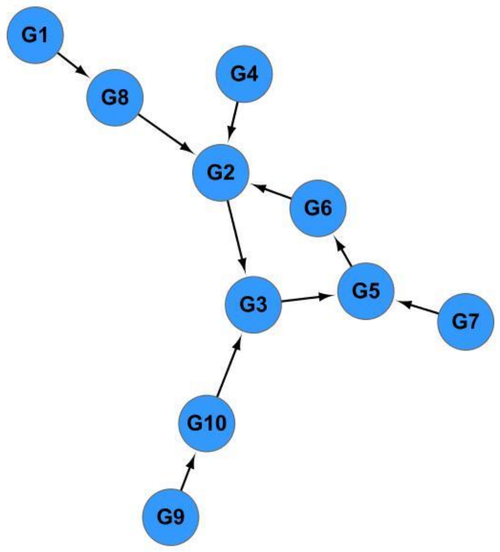
$$p(A,B) = \frac{w(A,B)}{\max\limits_{e \in E} w(e)} \tag{3.20}$$

where *w(A, B)* and *w(e)* present the weight of edge *(A, B)* in the set of edges *E*. Then they seed this prior network to bootstrapping. In each bootstrap iteration, the features of interest are checked such like directed edge, undirected edge, order relation, and Markov relation. The confidence of overall bootstrap can be evaluated by measuring how many times a specific feature appears related to the total number of iterations, which thus supports chosen features. They also collected interactive data from high-throughput yeast two hybrid protein-protein interaction (PPI) screenings.

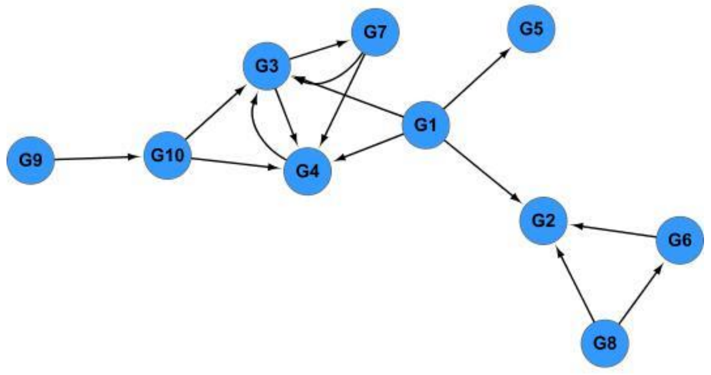
Haibe-Kains et al. (Haibe-Kains, Olsen et al. 2012) developed a web-based application called Predictive Networks (PN) to evaluate experimentally derived gene lists in the context of large-scale gene interaction networks. The PN analytical pipeline has two steps. At the beginning a comprehensive set of gene interactions extracted from a bunch of sources that are publicly available by applying text-mining algorithms. The second step consists of using these 'known' interactions together with gene expression data to infer robust gene networks by using supervised approaches, including regression and Bayesian methods. The PN web application can be accessed from http://predictivenetworks.org.

3.7 Some Preliminary Experimental Results

We evaluated the performance of some of the unsupervised gene network inference algorithms surveyed in this dissertation. The NIR algorithm was implemented in MATLAB 7 and run on a Linux platform. We tested the NIR algorithm using DREAM4 knock out data. Figure 3.1 shows the network predicted by the NIR algorithm, and the standard network, i.e., the ground truth, provided by DREAM4.



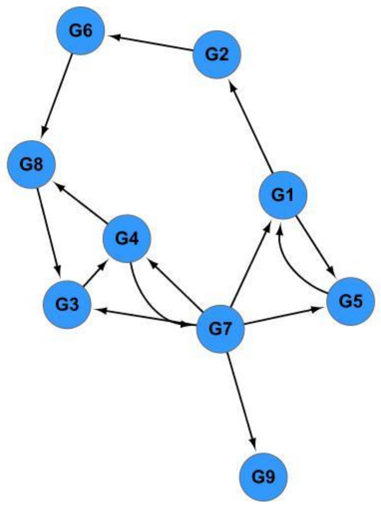
(A)



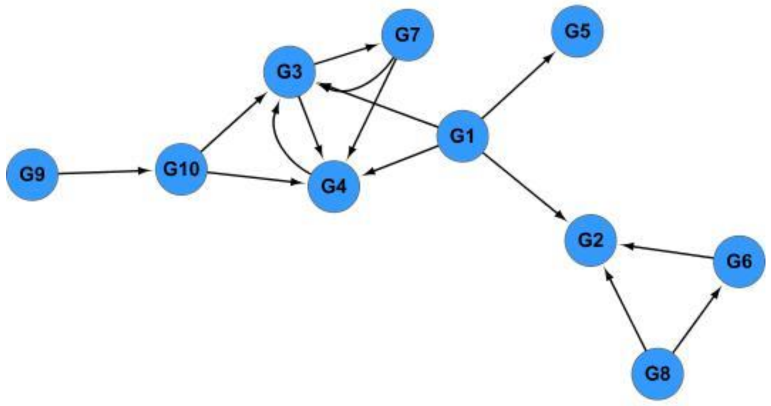
(B)

Figure 3.1 Results obtained by running NIR on DREAM4 knockout data. (A) The network predicted by the NIR algorithm. (B) The standard network from DREAM4.

The GENIE3 algorithm was implemented in R and run on a Windows platform. We tested the GENIE3 algorithm using DREAM4 knock out data. Figure 3.2 shows the network predicted by the GENIE3 algorithm and the standard network, i.e., the ground truth, provided by DREAM4.



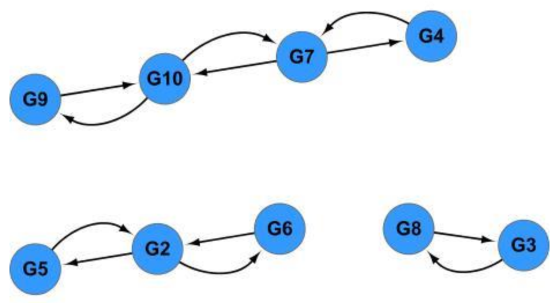
(A)



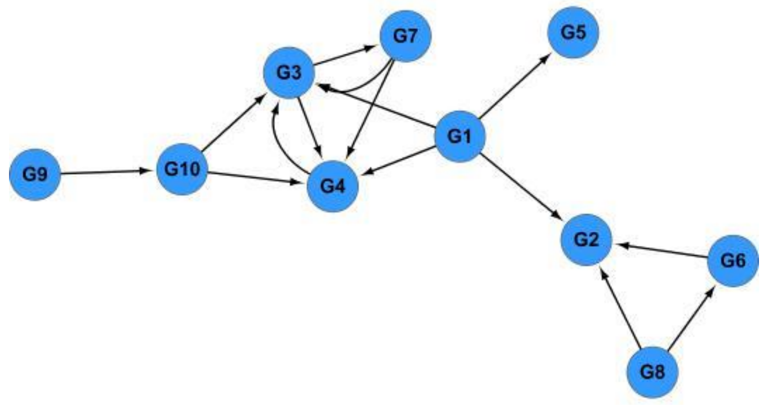
(B)

Figure 3.2 Results obtained by running GENIE3 on DREAM4 knockout data. (A) The network predicted by the GENIE3 algorithm. (B) The standard network from DREAM4.

The CLR algorithm was implemented in MATLAB 2011 and run on a Windows platform. We tested the CLR algorithm using DREAM4 knock out data. Figure 3.3 shows the network predicted by the CLR algorithm and the standard network, i.e., the ground truth, provided by DREAM4.



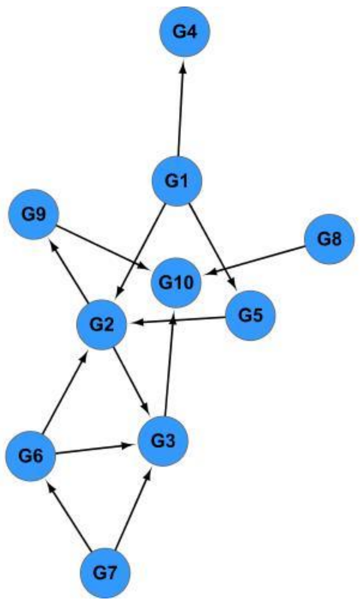
(A)



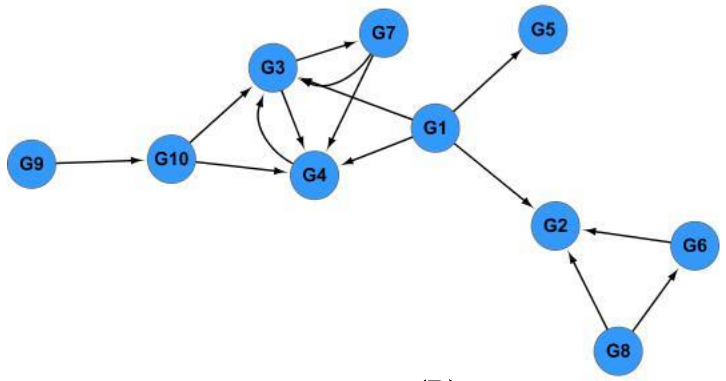
(B)

Figure 3.3 Results obtained by running CLR on DREAM4 knockout data. (A) The network predicted by the CLR algorithm. (B) The standard network from DREAM4.

The time-delay ARACNE, or TDARACNE, algorithm was implemented in R and run on a Windows platform. We tested the time-delay ARACNE algorithm using DREAM4 time series data. Figure 3.4 shows the network predicted by the TDARACNE algorithm and the standard network, i.e., the ground truth, obtained from DREAM4.



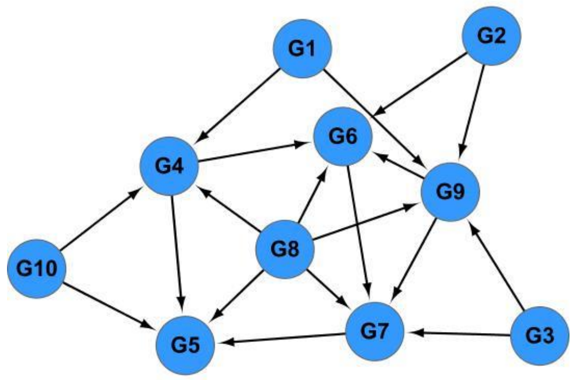
(A)



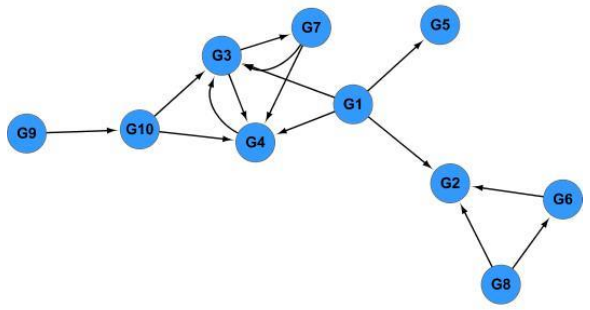
(B)

Figure 3.4 Results obtained by running TDARACNE on DREAM4 time series data. (A) The network predicted by the TDARACNE algorithm. (B) The standard network from DREAM4.

The Banjo algorithm was implemented in Java 7.0 and run on a Windows platform. We tested the Banjo algorithm using DREAM4 time series data. Figure 3.5 shows the network predicted by the Banjo algorithm and the standard network, i.e., the ground truth, obtained from DREAM4.



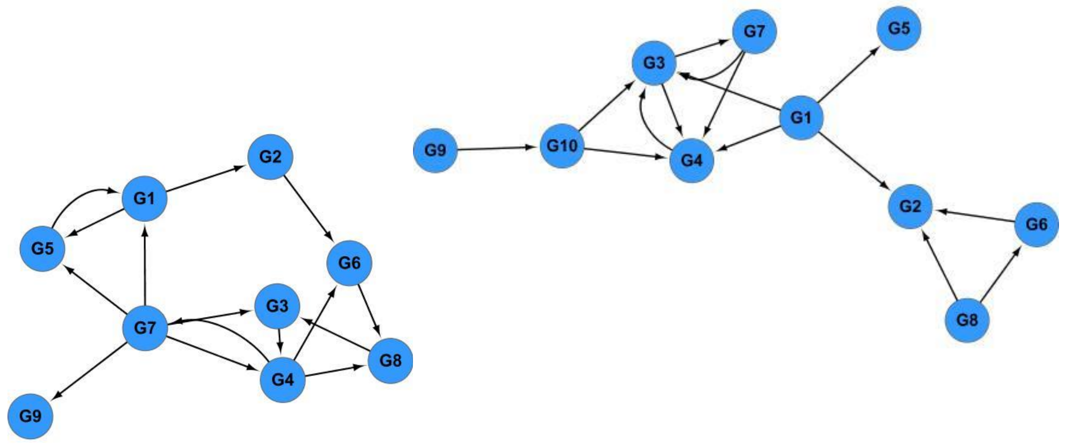
(A)



(B)

Figure 3.5 Results obtained by running Banjo on DREAM4 time series data. (A) The network predicted by the Banjo algorithm. (B) The standard network from DREAM4.

All the algorithms in the pipeline of Inferelator were implemented in R and run on a Linux platform. We tested Inferelator using both steady-state data (specifically, DREAM4 knockout data) and DREAM4 time series data. Figure 3.6 shows the network predicted by Inferelator and the standard network, i.e., the ground truth, obtained from DREAM4.



(A) **(B)**

Figure 3.6 Results obtained by running Inferelator on DREAM4 knockout data and time series data. (A) The network predicted by Inferelator. (B) The standard network from DREAM4.

In evaluating the performance of the network prediction algorithms, we use measures including accuracy, precision, and recall. We use TP (true positive) to denote the number of positive edges that are predicted correctly. A positive edge is one that appears in the ground truth where the ground truth is the standard network provided by

DREAM4. We use TN (true negative) to denote the number of negative edges that are predicted correctly. A negative edge is one that does not appear in the ground truth. We use FN (false negative) to denote the number of positive edges that are incorrectly predicted as negative. We use the FP (false positive) to denote the number of negative edges that are incorrectly predicted as positive. Then

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.21}$$

$$precision = \frac{TP}{TP + FP} \tag{3.22}$$

$$recall = \frac{TP}{TP + FN} \tag{3.23}$$

The accuracy, precision, and recall of the six tools tested in this dissertation proposal are shown in the following table.

Table 3.3 Accuracy, Precision, and Recall of Tested GRN Algorithms

Algorithm	**Accuracy**	**Precision**	**Recall**
Algorithms using steady-state data			
NIR	0.8	0.4	0.266667
GEINE3	0.786517	0.357143	0.333333
CLR	0.758621	0.25	0.2
WGCNA	0.756	0.267	0.267
Algorithms using time-series data			
TDARACNE	0.818182	0.461538	0.4
Banjo	0.702128	0.157895	0.2
Algorithms using both steady-state data and time series data			
Inferelator	0.786517	0.357143	0.333333

CHAPTER 4

A GPU PROGRAM FOR WHOLE GENOME COMPARISON

4.1 Introduction

A multiple sequence alignment (MSA) is related sets by partitioning residues such like amino acids or nucleotides from a given set of sequences. As the prerequisite for most phylogenetic and evolutionary analyses, MSA is the basic biological sequence analysis for the research of the relationship about evolutionary homology (Wallace, Blackshields et al. 2005) (Edgar and Batzoglou 2006) (Notredame 2007). Most of the MSAs attempt to align every residue in all the sequences, which is called "global alignments". The methods include mutational processes dealing with residue substitution, subsequence insertion, and subsequence deletion (Notredame 2007). After the whole-genome sequences has been discovered, it raises an interest in MSAs for whole genomes, which is called whole-genome alignment (WGA), covering all kinds of sequences: genes, promoters, repetitive regions, etc. This research requires more analyses about genome rearrangements, such like inversions, translocations, chromosome fusions, chromosome fissions, and reciprocal translocations. Meanwhile, another tools can also model unbalanced rearrangements which will copy number change, like tandem and segmental duplications (Blanchette, Kent et al. 2004, Miller, Rosenbloom et al. 2007, Paten, Herrero et al. 2008, Angiuoli and Salzberg 2011, Paten, Earl et al. 2011). The development for whole genome alignment is becoming more critical to figure out the selective forces acting across genomes, improve analysis of many potential functional elements such like the identification of conserved non-coding functional elements (Lindblad-Toh, Garber et al. 2011), along with cis-regulatory elements (Kellis, Patterson et al. 2003), enhancers, and non-coding RNAs (Earl, Nguyen et al. 2014).

Our work focuses on searching anchors, termed as non-gap alignment segments of local alignments, for whole genomes via multiple sequence alignments. To identify the commonalities and differences of two genomes from closely related organisms, for example, human and gorilla, various programs have been developed for sequencing alignment, applied to the whole genomes by using a seed-and-extend technique, beginning from exact or close exact matches and choosing a reliable subset of them, which is called anchors, then chaining all the anchors corresponding to the reference genome with overlapped or duplicated anchors removed and minimum of gaps remaining.

Several approaches for multiple sequence alignments to search anchors are available. We started with LASTZ (Harris 2007) and SSEARCH (Pearson 2000). LASTZ is a drop-in replacement for BLASTZ (Schwartz, Kent et al. 2003), which recognizes primary anchors as high-scoring local alignments before any other tools. The anchors start from pairs of spaced 12-mers with possibly one transition, and then are extended in two stages, if substitutions and gaps are allowed. The method is also applied for finding new, secondary matches between consecutive primary matches (Lippert, Zhao et al. 2005). On the other hand, SSEARCH does a rigorous Smith-Waterman search (Smith and Waterman 1981) for similarity between a query sequence and a group of sequences of the same type, i.e., chromosome or protein. This may be the most sensitive method available for similarity searches.

Compared to LASTZ, SSEARCH can be very slow, taking days and months to map millions of bases for mammalian genome sequences. This is the reason why we need to rely on programming in GPU (Graphics Processing Unit). The GPU is designed to run in parallel hundreds of short functions called threads, which are organized into blocks then in turn organized into grids. Parallel algorithms running on GPUs can often achieve up to 100x speedup over similar CPU algorithms. Our plan is to apply the GPU program via the algorithm of MaxSSmap (Turki and Roshan 2014) for the multiple sequence alignments with whole genome sequences to search anchors in evolutionary analysis, increasing accuracy and speed of processing at the same time.

4.2 Dataset

The dataset used to test our alignment program via GPU is extracted from a competitive assessment of whole genome alignment methods named Alignathon (Earl, Nguyen et al. 2014). Alignathon used three test sets, two of them were simulated datasets created by forward-time simulation with the EVOLVER tool (Edga, Asimeno et al. 2009). The first set models a phylogeny of great ape containing the genomes such as humans, chimpanzees, gorillas, and orangutans, all with the same evolutionary relationships. Our program tested tentatively two genomes from this set, humans and gorillas. No doubt that the outcome has impressively high accuracy, above 0.8 and 0.9 respectively in precise and recall. But our research focuses on distant related organisms; in consequence the major dataset we used for our program is the second simulated dataset from Alignathon, which is about a mammalian phylogeny containing genomes as humans, mice, rats, cows, and dogs. We chose two distant related organisms, cow and mouse, from this set. The two genomes that are mainly tested in our program are the chromosome C from species cow and the chromosome O from species mouse. The phylogenetic distance between these two genomes is 0.60 (Earl, Nguyen et al. 2014).

4.3 Implementation and Methods

The algorithm used for the genome alignment in our research is called MaxSSmap (Turki and Roshan 2014). The running platform is CUDA 6.0 implementing on NVIDIA GPUs. The GPU, as graphics processing unit, is designed to execute hundreds of small functions called threads at the same time in parallel. All the threads are bundled into blocks which then are organized into grids. While the MaxSSmap is running, only one grid are applied, and the number of blocks is set to the total number of fragments, the contiguous parts with constant length defined by the user, from the reference genome sequence, normally the cow genome in our research. The number of fragments is determined by how many threads in a block are executed simultaneously. By default the value is set to 256.

The query genome sequence, normally the mouse genome, is divided into short length reads. The length of the reads is not necessary to be constant. The input of the MaxSSmap program is the whole reference genome and a read. The program has two steps. The first to identify a local region of the reference genome, in our case is to locate the fragment ID number by sliding the short read through the whole reference genome and picking up the maximum scoring subsequence (Bates and Constable 1985, Bentley 1986). The maximum scoring subsequence is defined to maximize the sum of a region in the original sequence. For instance, the original sequence contains a list of real numbers $\{x_1, x_2, ..., x_n\}$, and the maximum scoring subsequence should be the contiguous subsequence $\{ x_i, ..., x_j \}$ whose sum $x_i + ... + x_j$ $(0 \leq i, j \leq n)$ is maximized. For DNA sequences, the list of real numbers is replaced as the cost score list under two aligned sequences with same length and without gaps. The cost scores can be defined from a position specific scoring matrix presenting base call probabilities, or a substitution scoring matrix, or a trivial match or mismatch cost. In Figure 4.1 we show up a brief overview of the first phase of MaxSSmap program. Each thread of the GPU is input the read and one fragment of the reference genome, and each fragment is assigned with an ID number, from 0, 1, 2, ... and so on. In each thread the read is sliding through the fragment and the maximum scoring subsequence is computed. To consider the cross junctions between fragments, the neighboring fragments are also included to map the read in each thread. The output of the execution is the fragment numbers with the highest and second highest scores. With the assistant of the second highest score, redundant false positives can be removed if we set up a threshold of the ratio of the second highest score and the first highest score.

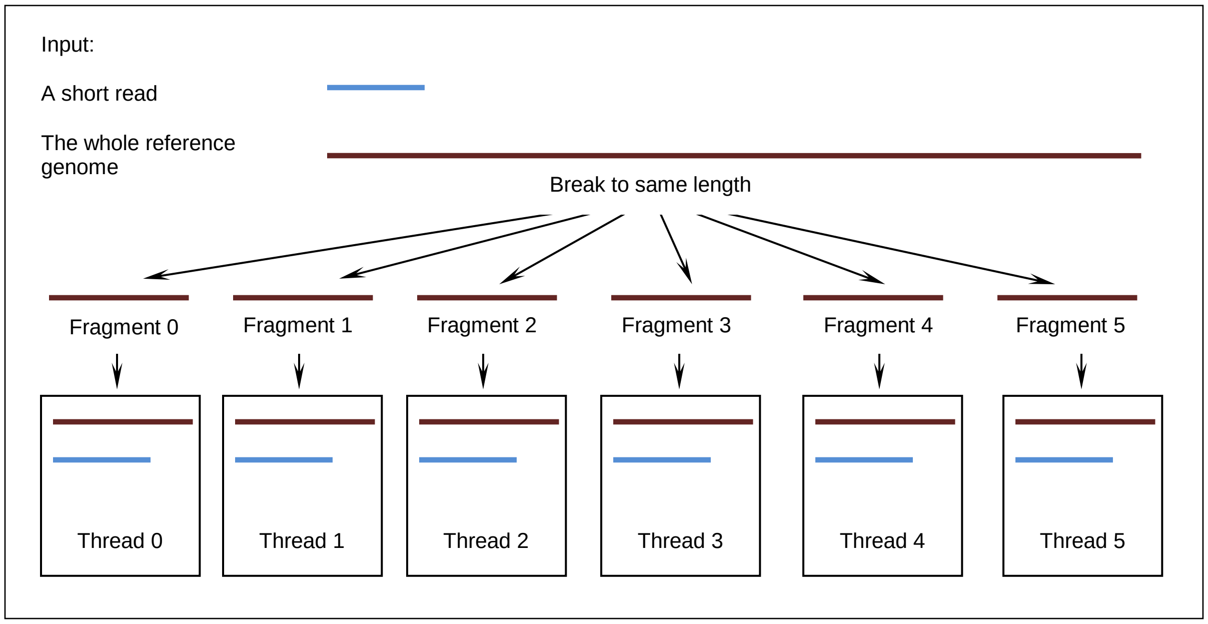


Figure 4.1 First phrase of MaxSSmap. The whole reference genome is divided to six same size fragments with ID numbers from 1 to 6 and fed into six threads of the GPU. Each thread will execute with one fragment and the short read, sliding the read with the fragment and looking for the maximum scoring subsequence. The read is also mapped to the junctions between fragments to make sure that the read is fully mapped to the reference.

Source: Turki, T. and U. Roshan (2014). "MaxSSmap: a GPU program for mapping divergent short reads to genomes with the maximum scoring subsequence." BMC Genomics **15**: 969.

The second step of MaxSSmap program is applying a gap-allowed alignment method to align the read with the region of the reference genome starting at the identified fragment picked up from the first step. To obtain enough nucleotides as the read to achieve the alignment, spanning the fragments to the right is necessary at most time. Furthermore, we tested three algorithms in this phrase, Needleman-Wunsch (Needleman and Wunsch 1970), Smith-Waterman (Smith and Waterman 1981), and the extention for the Smith-Waterman approach, by feeding the outputs of local alignments to an genome-wide mammalian consistency based alignment method named Pecan (Paten, Herrero et al. 2008), to pursuit for higher accuracy for computing genome aligments. In the rest context of this paper, we will elaborate in detail the three alignment methods, their results from our experiments with the two mammlian genome sequences, cow chromosome C and mouse chromsome O, and the analyses and comparison among the three algorithms and other published alignment tools.

4.3.1 Needleman-Wunsch Algorithm

The algorithm of Needleman-Wunsh (Needleman and Wunsch 1970) is extensively applied for finding similarities and determining whether significant homology exists between nucleotide or protain sequences. Although the method was originally published in more than 40 years ago, it is still widely used in recent eras for optimal global alignment, particularly when the quality of the global alignment is of the utmost importance. It was one of the first applications of dynamic programming to compare biological sequences. The basic idea of the algorithm is to build up the best alignment by using optimal alignments of smaller subsequences, meanwhile to reduce the massive number of possibilities that need to be considered, yet still guarantees that the best solution will be found. Based on a divide and conquer strategy, the algorithm consists of the following steps:

1. Divide the problem into smaller sub problems. In the alignment algorithm, we break the sequences to be aligned to based pairs, which also contain the gapped alignments.
2. Solve the smaller problems optimally. The scores of all the probabilities of base pairs are computed and stored in the trace back table. We trace the base pairs from the end of the two sequences, and define the optimal alignments according to the best scores.
3. Use the sub-problem solutions to construct an optimal solution for the original problem. By tracing the optimal base pairs (including gaps) step by step, the final alignment with maximum match and best score is obtained.

The basic mathematic equation for Needleman-Wunsh algorithm is shown as follow:

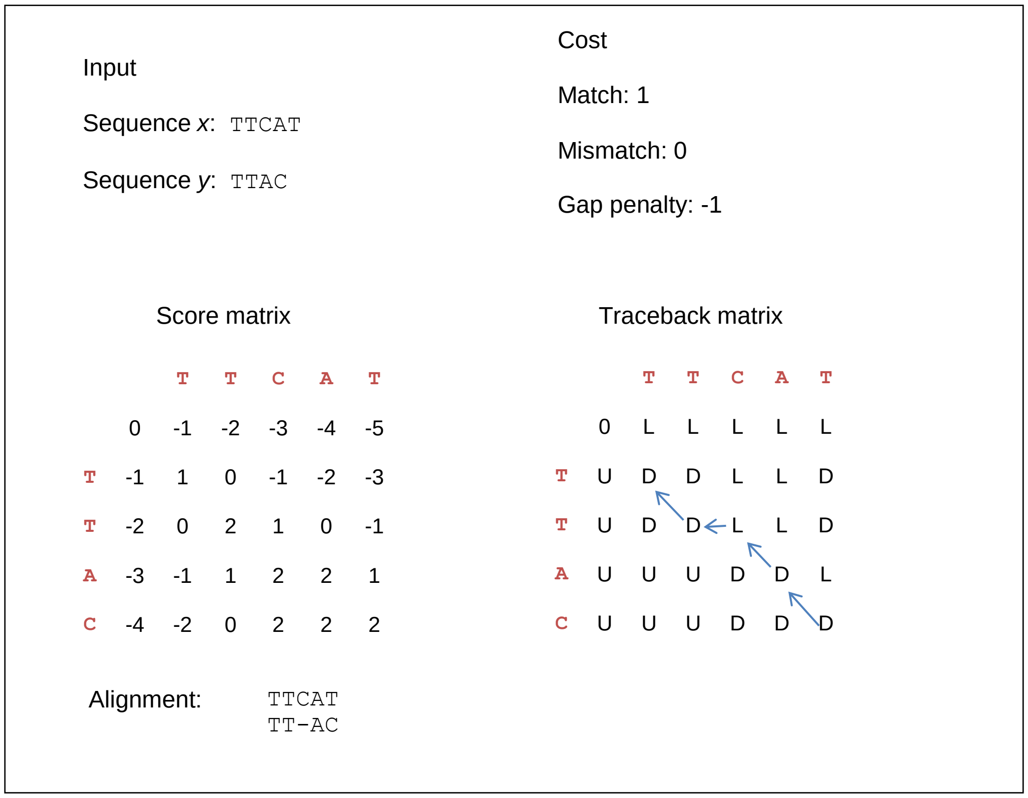
$$D(i,j) = max \begin{cases} D(i-1,j-1) + s(x_i, y_j) \\ \quad D(i-1) + g \\ \quad D(j-1) + g \end{cases} \tag{4.1}$$

The equation 4.1 helps us to create recursively the trace back matrix $D(i, j)$ indexed by the residues of two sequences *x* and *y*, with a boundry condition such that:

$$\begin{aligned} D(i,0) &= g \times i \\ D(0,j) &= g \times j \end{aligned} \tag{4.2}$$

Where *g* is the gap penalty. The substitutuon score $s(x_i, y_j)$ is for the residues *i* and *j* in the two sequences *x* and *y* respectively.

Therefore, we need two matrix tables for the trace back process. One is to store the maximum scores calculated by the equation 4.1, the other is for the trace back records, containing the information how the maximum scores are obtained: 1) two residues align together, the case named "diagonal", or "match/mismatch", 2) a gap is inserted in the sequence *x*, the case named "up", in some literatures it is also called "deletion", 3) a gap is inserted in the sequence y, the case named "left", or "insertion". A trival example presenting in Figure 4.2 will illuminate the exact procedures how to optimize an alignment.



Input

Sequence *x*: `TTCAT`

Sequence *y*: `TTAC`

Cost

Match: 1

Mismatch: 0

Gap penalty: -1

Score matrix

$$\begin{array}{ccccccc}
 & & T & T & C & A & T \\
 & 0 & -1 & -2 & -3 & -4 & -5 \\
T & -1 & 1 & 0 & -1 & -2 & -3 \\
T & -2 & 0 & 2 & 1 & 0 & -1 \\
A & -3 & -1 & 1 & 2 & 2 & 1 \\
C & -4 & -2 & 0 & 2 & 2 & 2
\end{array}$$

Traceback matrix

$$\begin{array}{ccccccc}
 & & T & T & C & A & T \\
 & 0 & L & L & L & L & L \\
T & U & D & D & L & L & D \\
T & U & D & D \leftarrow & L & L & D \\
A & U & U & U & D & D & L \\
C & U & U & U & D & D & D
\end{array}$$

Alignment:

```
TTCAT
TT-AC
```

Figure 4.2 Procedures of the algorithm of Needleman-Wunsh. Given the input of two sequences *x* and *y*, the score matrix is calculated via equations 3.1and 3.2. The way to obtained the maximum value of each cell in the score matrix is stored in the traceback matrix: diagonal (maked as D), up (marked as U), or left (marked as L). By tracing back from the lower right corner of the traceback matrix, the optimal alignment is built up.

4.3.2 Smith-Waterman Algorithm

Another computing method for local sequence alignment was also tested in our GPU program in the second phrase of MaxSSmap. This is an method called Smith-Waterman Algorithm (Smith and Waterman 1981) for comparison similiar to the algorithm of Needleman-Wunsh, but the difference between them is that the Smith-Waterman Algorithm performs local sequence alignment instead of global alignment consisting of the whole input sequences; that is, for determining similar regions between two strings or nucleotide or protein sequences. Hence, only part of the string of each input sequence is contained in the output aligment. The mathematic expression for the Smith-Waterman Algorithm is presenting as below:

$$M(i,j) = max\begin{cases} M(i-1,j-1) + s(x_i, y_j) \\ M(i-1) + g \\ M(j-1) + g \\ 0 \end{cases} \tag{4.3}$$

with a distinct initial condition from the algorithm of Needleman-Wunsh:

$$\begin{aligned} M(i,0) &= 0 \\ M(0,j) &= 0 \end{aligned} \tag{4.4}$$

Comparing the equation 4.3 with the equation 4.1, one more condition is appended to calculate the optimal similarity by setting up a zero to prevent calculating negative similarity which indicates that no similarity up to x_i and y_j. The alogrithm yields an alignment consisting of a series of local alignments with optimal similarities in specific regions of the two input sequences, which have not any association among them.

4.3.3 Smith-Waterman Extended with Pecan

The algorithm of Pecan was developed as a tool for large-scale probabilitic consistency alignment (Paten, Herrero et al. 2008). It implements the similar basic objective function as the tool of Probcons (Do, Mahabhashyam et al. 2005), an animo acid aligner. The basic idea of these two approches is derived from the concept of a pair-hidden Markov model (pair-HMM) (Durbin 1998). Figure 4.3 shows the conceptual graph of pair-HMM that specifies the probability distribution over all alignments between a pair of sequences.

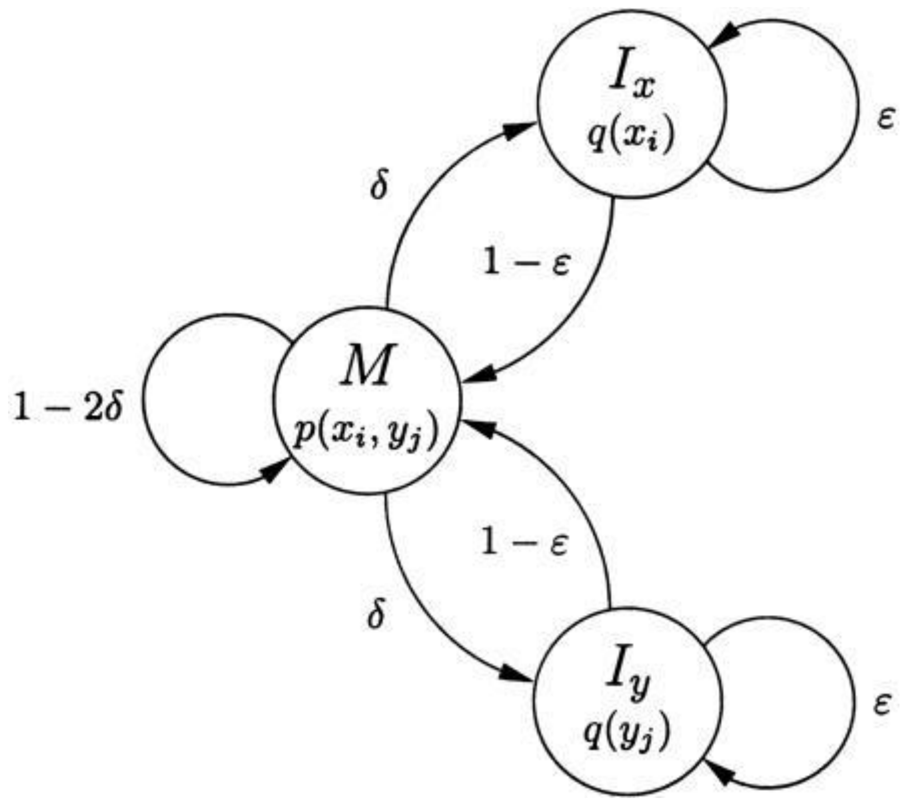


Figure 4.3 Basic pair-HMM for sequence alignment for two sequences, *x* and *y*. State *M* emits two residues, x_i and y_j, respectively from the two sequences, presenting the two residues being aligned together. State I_x emits a residue in sequence *x* aligned to a gap, and similarly state I_y emits a residue in sequence *y* aligned to a gap too. The optimal similarity is obtained by applying Needleman-Wunsch algorithm with suitable parameters. The emission probability function *p*(.,.) at state *M* corresponds to a substitution scoring matrix, at the same time affine gap penalty parameters can be derived from the transition probabilities δ and ε (Durbin 1998).

Source: Do, C. B., M. S. Mahabhashyam, et al. (2005). "ProbCons: Probabilistic consistency-based multiple sequence alignment." Genome Res **15**(2): 330-340.

The algorithm Pecan hace four main phrase:

1. Create a "constraint map" consisting of a set of alignment constraints that satisfy that the two residues x_i and y_j has a constraint $i < j$.

2. Calculate a set of pairwise posterior match probabilities according to the constraint map created in step 1.

3. Modify the set of posterior match probabilities using the consistency transformed with the reference from a third sequence *z* out of the group.

4. Combine the transformed posterior match probabilities into a multiple alignment by applying a method of progressive alignment.

By the reason that only two genome sequences are going to be compared in our GPU program, we ignore the last two steps of Pecan. The original source to create a constraint map in the first phrase is a set of alignment anchors which are continuous un-gapped series of one or more aligned pairs, produced by using Exonerate (Slater and Birney 2005) in the original Pecan program. We modified the Pecan scripts and computed the constraint map with the local alignments yielded from the Smith-Waterman algorthm.

The objective to create the constraint map is to gain more efficiency for the program by avoiding conflicts among the anchor chain and choosing a colinear and non-overlapping chain of anchor constraints to construct an alignment band. Based on the alignment band the posterior match probablitlies are calculated by using backward algorithm to set up the pair-hidden Markov model (Paten, Herrero et al. 2008). In order to describe the backward calculation for pair HMM to produce posterior match probablitlies, we introduce a new notation $x_i \lozenge y_j$, which means that x_i is aligned to y_j. Then based on the standard conditional probability therory we have

$$\begin{aligned} P(x, y, x_i \lozenge y_j) &= P(x_{1\ldots i}, y_{1\ldots j,} x_i \lozenge y_j) P(x_{i+1\ldots n}, y_{j+1\ldots m} | x_{1\ldots i}, y_{1\ldots j,} x_i \lozenge y_j) \\ &= P(x_{1\ldots i}, y_{1\ldots j,} x_i \lozenge y_j) P(x_{i+1\ldots n}, y_{j+1\ldots m} | x_i \lozenge y_j) \end{aligned} \tag{4.5}$$

where n is the length of sequence x, and m is the length of sequence y, with $0 \le i \le n$ and $0 \le j \le m$. The first term of equation 4.5 is the forward probability while the second term is the corresponding probability $b^M(i, j)$ which can be calculated via backward algorithm as shown in Figure 4.4.

```
Algorithm: Backward calculation for pair HMMs
Initialization:
    $b^M(n,m) = b^X(n,m) = b^Y(n,m) = \tau$
    All $b^*(i,m+1), b^*(n+1,j)$ are set to 0.
Recursion: $i = n, \dots 1, j = m, \dots ,1$ except $(n,m)$
    $b^M(i,j) = (1-2\delta-\tau)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1)$
            $+ \delta[q_{x_{i+1}}b^X(i+1,j) + q_{y_{j+1}}b^Y(i,j+1)]$
    $b^X(i,j) = (1-\varepsilon-\tau)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \varepsilon q_{x_{i+1}}b^X(i+1,j)$
    $b^Y(i,j) = (1-\varepsilon-\tau)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \varepsilon q_{y_{i+1}}b^Y(i+1,j)$
```

Figure 4.4 Backward Algorithm for calculation of posterior match probablitlies. No special termination step is needed, since only the values of $b^*(i,j)$ where $i,j \geq 1$ are required to calculate.

Source: Durbin, R. (1998). Biological sequence analysis : probabalistic models of proteins and nucleic acids. Cambridge, UK New York, Cambridge University Press.

Then we can use Bayes theorem to obtain

$$P\left(x_i \lozenge y_j \middle| x,y\right) = \frac{P(x,y,x_i \lozenge y_j)}{P(x,y)} \tag{4.6}$$

And similar values for the posterior probabilities of using specific insert states (the anchor chain in our program) can also be produced.

4.4 Results

Due to the test set for out GPU program is from the Alignathon assessment, we also applied the same comparison tool to test the results. All the output files were transferred to multiple alignment format (MAF), and compared with the simulated truth file provided from the project Alignathon. Given two input MAF files, the comparator tool calculates the precision, recall, and F-score, which is a standard method for combining precision and recall into a single value, as shown in equation 4.7:

$$Fscore = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.7}$$

As described in the section of implement and methods, we implemented three algorithms for our GPU program for genome comparison. The first one is MaxSSmap with similarities computing with the Needleman-Wunsch algorithm, the second is MaxSSmap with the comparison procedure executed with Smith-Waterman algorithm, and the last on is the extension of the second approach, by importing the output of the MaxSSmap with Smith-Waterman to the Pecan algorithm, and calculating a better comparison. In this section we list all the results from these three methods, with various parameters, such like read length, the step to extract reads from query genome (the mouse genome in our experiments), the overlap of each read and the one following it, etc.

Table 4.1 Results from MaxSSmap with Needleman-Wunsch, Various in Accordance with Different Length of Reads

Reference genome: simCow.chrC, 33408597 bases
Query genome: simMouse.chrO, 3949899 bases

Read Length	Step	True Positive	False Positive	False Negative	Precision	Recall	F-score
1000	100	41200	343642	1828051	10.7%	2.2%	0.036
500	100	121906	464123	1767345	20.8%	6.5%	0.099
250	100	238906	540904	1650904	30.6%	12.6%	0.179
200	50	298963	888271	1590288	25.2%	15.8%	0.194
150	50	295916	1013809	1593635	22.6%	15.6%	0.185
100	50	233729	979345	1655522	19.3%	12.4%	0.151
50	25	149589	1772374	1739662	7.8%	7.9%	0.079

The results from Table 4.1 shows up the performance of the algorithm MaxSSmap with Needleman-Wunsch approach which calculate the similarities in the second phrase. The longest read length yields the less false positive and the lowest F-score. And there is a peak of F-score with a specific read length of 200. This trend is also graphed in Figure 4.5.

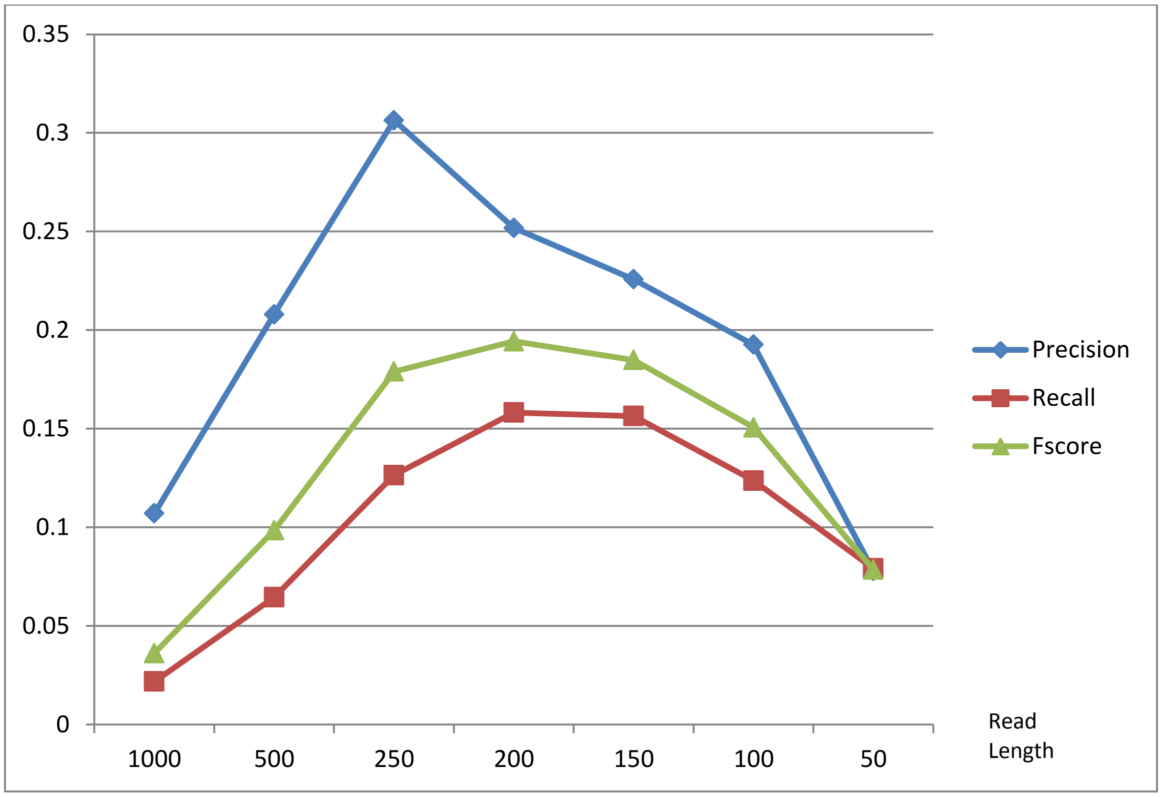


Figure 4.5 Trend of results from MaxSSmap with Needleman-Wunsch with different read length.

Therefore the most of rest experiments for the GPU program are based on the read length of 200. Table 4.2 presents all the results of MaxSSmap with Needleman-Wunsch based on a list of ascending steps, the intervals to cut reads from the query genome. The Figure 4.6 presents the trend of precision, recall, and F-score of all these experiments.

Table 4.2 Results from MaxSSmap with Needleman-Wunsch on Read Length of 200

Read Length	Step	Overlap	True Positive	False Positive	False Negative	Precision	Recall	F-score
200	5	195	165679	302919	1723572	35.4%	8.8%	0.141
200	10	190	175227	305508	1714024	36.5%	9.3%	0.148
200	15	185	174917	319453	1714334	35.4%	93%	0.147
200	20	180	182322	329623	1706929	35.6%	9.7%	0.152
200	25	175	184720	330369	1704531	35.9%	9.8%	0.154
200	30	170	190208	343733	1699043	35.6%	10.1%	0.157
200	35	165	193450	356178	1695801	35.2%	10.2%	0.159
200	40	160	196868	365196	1692383	35.0%	10.4%	0.161
200	45	155	203448	377316	1685803	35.0%	10.8%	0.165
200	50	150	210434	387820	1678817	35.2%	11.1%	0.169
200	55	145	217764	402344	1671487	35.1%	11.5%	0.174
200	60	140	222479	415696	1666772	34.9%	11.8%	0.176
200	65	135	226616	436323	1662635	34.2%	12.0%	0.178
200	70	130	229272	441390	1659979	34.2%	12.1%	0.179
200	75	125	228393	452800	1660858	33.5%	12.1%	0.178
200	80	120	238191	475023	1651060	33.4%	12.6%	0.183
200	85	115	239737	499613	1649514	32.4%	12.7%	0.182
200	90	110	244804	510741	1644447	32.4%	13.0%	0.185
200	95	105	250658	525303	1638593	32.3%	13.3%	0.188
200	100	100	251922	548718	1637329	31.5%	13.3%	0.187
200	105	95	261394	571980	1627857	31.4%	13.8%	0.192
200	110	90	265090	599648	1624161	30.7%	14.0%	0.193
200	115	85	266187	623076	1623064	29.9%	14.1%	0.192
200	120	80	268965	652207	1620286	29.2%	14.2%	0.191
200	125	75	277157	688233	1612094	28.7%	14.7%	0.194
200	130	70	279334	715235	1609917	28.1%	14.8%	0.194
200	135	65	286987	757896	1602264	27.5%	15.2%	0.196
200	140	60	289490	796505	1599761	26.7%	15.3%	0.195
200	145	55	294021	843710	1595230	25.8%	15.6%	0.194
200	150	50	298963	888271	1590288	25.2%	15.8%	0.194
200	155	45	307068	957968	1582183	24.3%	16.3%	0.195
200	160	40	312467	1019601	1576784	23.5%	16.6%	0.194
200	165	35	317541	1095894	1571710	22.5%	16.8%	0.192
200	170	30	324668	1199881	1564583	21.3%	17.2%	0.190
200	175	25	331634	1314967	1557617	20.1%	17.6%	0.188
200	180	20	339605	1469755	1549646	18.8%	18.0%	0.184
200	185	15	349260	1685460	1539991	17.2%	18.5%	0.178
200	190	10	360624	2001187	1528627	15.3%	19.1%	0.170
200	195	5	377133	2583476	1512118	12.7%	20.0%	0.156

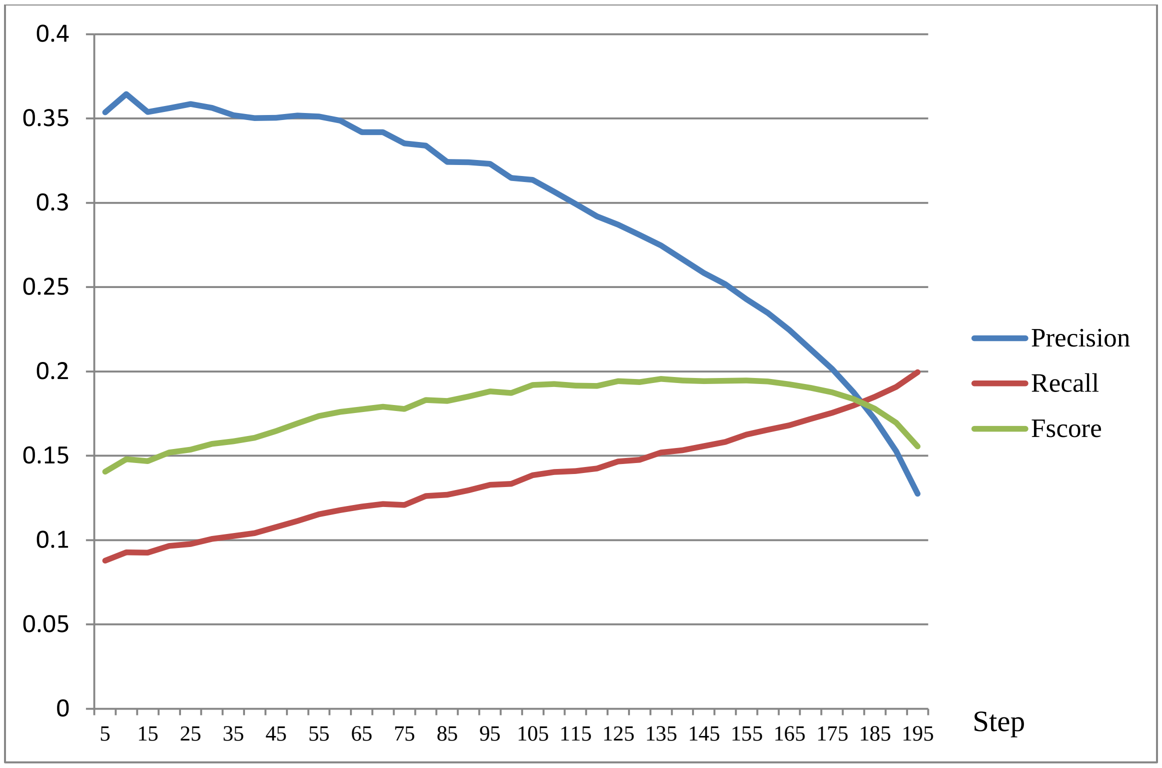


Figure 4.6 Trend of results from MaxSSmap with Needleman-Wunsch with read length fixed at 200.

From Table 4.2 and Figure 4.6, we can find the peak of F-score is produced with the step 135. In consequence, the group of parameters, read length of 200 and step of 135, is also the major concern for the next two program implementing with Smith-Waterman and pecan, along with the parameters group of read length of 100 and step 0f 100, and the one of 1000 and 100 in read length and step, respectively.

Table 4.3 presents the results from MaxSSmap with Smith-Waterman in the second phase. The output of Smith-Waterman for each read has two alignments companied with two scores, the best similarity and the second. And we set up a threshold of the ratio of the best score and the second one, to reduce the false positives due to repeats. Table 4.3 shows all the results with and without the threshold.

Table 4.3 Results from MaxSSmap with Smith-Waterman

Read Length	Step	Overlap	True Positive	False Positive	False Negative	Precision	Recall	F-score	Ratio
200	135	95	393589	4098681	1495662	8.8%	20.8%	0.123	n/a
200	135	95	310226	684460	1579025	31.2%	16.4%	0.215	0.9
500	400	100	373365	1921706	1515886	16.3%	16.4%	0.164	n/a
500	400	100	303272	371965	1585979	44.9%	16.1%	0.237	0.9
1000	900	100	273458	784081	1615793	25.9%	14.5%	0.186	n/a
1000	900	100	230524	183387	1658727	55.7%	12.2%	0.200	0.9

And the comparison of precision, recall, and F-score depending on the three parameter groups is presented in Figure 4.7.

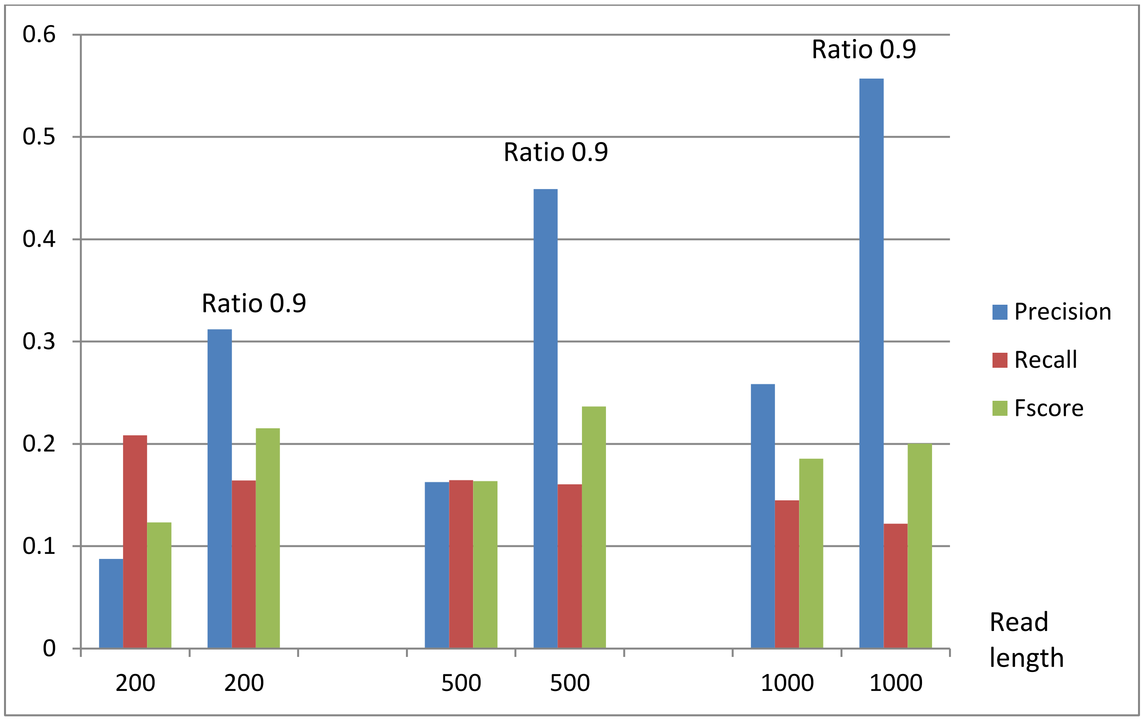


Figure 4.7 Comparison of outputs from MaxSSmap with Smith-Waterman.

From Table 4.3 and Figure 4.7, the most recent best F-score is yielded with the parameter group consisting read length at 500 and step at 100, along with the threshold of the ratio of best and second scores set up at 0.9.

To pursuing better performance, we import these output from MaxSSmap and Smith-Waterman to the algorithm of Pecan, replacing the exonerate execution in the original procedures. Table 4.4 presents the results from the amended Pecan with MaxSSmap.

Table 4.4 Results from Amended Pecan with MaxSSmap

Read Length	Step	Overlap	True Positive	False Positive	False Negative	Precision	Recall	F-score	Ratio
500	400	100	335203	286083	1554048	54.0%	17.7%	0.267	n/a
500	400	100	345714	206787	1543537	62.6%	18.3%	0.283	0.9
500	400	100	344852	207077	1544399	62.5%	18.3%	0.283	0.85
500	400	100	346935	194771	1542316	64.0%	18.4%	0.285	0.8
500	400	100	344117	186397	1545134	64.9%	18.2%	0.284	0.75
500	400	100	343201	174552	1546050	66.3%	18.2%	0.285	0.7

And the graph presents the trend of the outputs of Pecan according to different ratios:

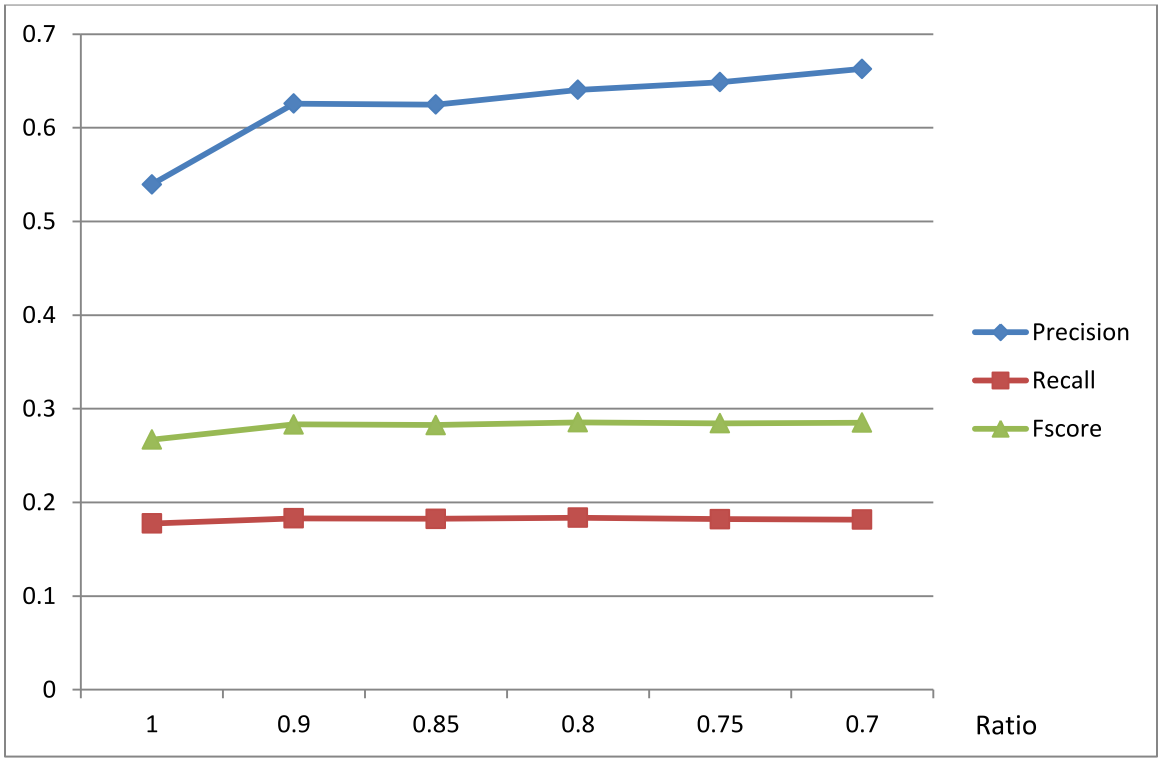


Figure 4.8 Trend of the outputs of Pecan according to different ratios.

Finally, we compare our best results with the published algorithm with the same input in Table 4.5.

Table 4.5 Results Compared with Published Algorithm

	True Positive	False Positive	False Negative	Precision	Recall	F-score
GPU	346935	194771	1542316	64.0%	18.4%	0.285
Pecan	294888	294157	1594363	50.1%	15.6%	0.238
Lastz	595362	270983879	1293889	2.2%	31.5%	0.005
Exonerate	114420	7118484	1774831	1.6%	6.1%	0.025

4.5 Discussion

From the experimental results demonstrated in the last section, we can conclude that the GPU program combined with the Smith-Waterman and Pecan has a better performance than other previously published algorithms. The F-score of GPU is more than 20% higher than Pecan, and much better than the two comparison tools: Lastz and Exonerate. The reason we chose the two input genomes, cow and mouse, is that the two organisms from the Alignathon test sets have the largest phylogenetic distance at 0.6. The outcomes from the experiments exhibit that the GPU program combined with efficient alignment tools has a strong capacity to calculate the similarities of distant related species.

Even for the first two tools we tested in our experiments, the MaxSSmap with Needleman-Wunsch, and with Smith-Waterman, higher F-scores yields than the two previous published alignment tools, Lastz and Exonerate. The GPU program can not only execute with efficient time consuming, but also produce competitive performance in genome alignment.

REFERENCES

An, J., Lai, J., Lehman, M. L. and Nelson, C. C. (2013). "miRDeep*: an integrated application tool for miRNA identification from RNA sequencing data." *Nucleic Acids Res* **41**(2): 727-737.

Anderson, M. and Schrijver, I. (2010). "Next generation DNA sequencing and the future of genomic medicine." *Genes* **1**(1): 32-69.

Angiuoli, S. V. and Salzberg, S. L. (2011). "Mugsy: fast multiple alignment of closely related whole genomes." *Bioinformatics* **27**(3): 334-342.

Aukerman, M. J. and Sakai, H. (2003). "Regulation of flowering time and floral organ identity by a MicroRNA and its APETALA2-like target genes." *Plant Cell* **15**(11): 2730-2741.

Bartel, D. P. (2004). "MicroRNAs: genomics, biogenesis, mechanism, and function." *Cell* **116**(2): 281-297.

Bates, J. L. and Constable, R. L. (1985). "Proofs as programs." *ACM Trans Program Land Syst* **7**: 113-136.

Bentley, J. L. (1986). *Programming pearls*. Reading, Mass., Addison-Wesley.

Bentwich, I., Avniel, A., Karov, Y., Aharonov, R., Gilad, S., Barad, O., Barzilai, A., Einat, P., Einav, U., Meiri, E., Sharon, E., Spector, Y. and Bentwich, Z. (2005). "Identification of hundreds of conserved and nonconserved human microRNAs." *Nat Genet* **37**(7): 766-770.

Bindewald, E. and Shapiro, B. A. (2006). "RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers." *RNA* **12**(3): 342-352.

Bindra, R. S., Wang, J. T. L. and Bagga, P. S. (2010). "Bioinformatics methods for studying microRNA and ARE-mediated regulation of post-transcriptional gene expression." *International Journal of Knowledge Discovery in Bioinformatics* **1**(3): 97-112.

Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D. and Miller, W. (2004). "Aligning multiple genomic sequences with the threaded blockset aligner." *Genome Res* **14**(4): 708-715.

Breiman, L. (1996). "Bagging predictors." *Machine Learning* **24**: 123-140.

Breiman, L. (2001). "Random Forests." *Machine Learning* **45**(1).

Brennecke, J., Hipfner, D. R., Stark, A., Russell, R. B. and Cohen, S. M. (2003). "bantam encodes a developmentally regulated microRNA that controls cell proliferation and regulates the proapoptotic gene hid in Drosophila." *Cell* **113**(1): 25-36.

Bushati, N. and Cohen, S. M. (2007). "microRNA functions." *Annual Review of Cell and Developmental Biology* **23**: 175-205.

Butte, A. J. and Kohane, I. S. (2000). "Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements." *Pac Symp Biocomput*: 418-429.

Cerulo, L., Elkan, C. and Ceccarelli, M. (2010). "Learning gene regulatory networks from only positive and unlabeled data." *BMC Bioinformatics* **11**: 228.

Cortes, C. and Vapnik, V. (1995). "Support-vector networks." *Machine Learning* **20**: 273-297.

Djebbari, A. and Quackenbush, J. (2008). "Seeded Bayesian Networks: constructing genetic networks from microarray data." *BMC Systems Biology* **2**: 57.

Do, C. B., Mahabhashyam, M. S., Brudno, M. and Batzoglou, S. (2005). "ProbCons: Probabilistic consistency-based multiple sequence alignment." *Genome Res* **15**(2): 330-340.

Durbin, R. (1998). *Biological sequence analysis : probabalistic models of proteins and nucleic acids*. Cambridge, New York: Cambridge University Press.

Earl, D., Nguyen, N., Hickey, G., Harris, R. S., Fitzgerald, S., Beal, K., Seledtsov, I., Molodtsov, V., Raney, B. J., Clawson, H., Kim, J., Kemena, C., Chang, J. M., Erb, I., Poliakov, A., Hou, M., Herrero, J., Kent, W. J., Solovyev, V., Darling, A. E., Ma, J., Notredame, C., Brudno, M., Dubchak, I., Haussler, D. and Paten, B. (2014). "Alignathon: a competitive assessment of whole-genome alignment methods." *Genome Res* **24**(12): 2077-2089.

Edga, R., Asimeno, G., Batzoglou, S. and Sidow, A. (2009). "EVOLVER. ." http://www.drive5.com/evolver/ (accessed on May 5, 2016).

Edgar, R. C. and Batzoglou, S. (2006). "Multiple sequence alignment." *Current Opinion in Structural Biology* **16**(3): 368-373.

Elati, M., Neuvial, P., Bolotin-Fukuhara, M., Barillot, E., Radvanyi, F. and Rouveirol, C. (2007). "LICORN: learning cooperative regulation networks from gene expression data." *Bioinformatics* **23**(18): 2407-2414.

Elkan, C. and Noto, K. (2008). "Learning classifiers from only positive and unlabeled data." *KDD'08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA:ACM* **2008**: 213-220.

Faith, J. J., Hayete, B., Thaden, J. T., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J. J. and Gardner, T. S. (2007). "Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles." *PLoS Biol* **5**(1): e8.

Fan, R., Chen, P. and Lin, C. (2005). "Working set selection using the second order information for training SVM." *Journal of Machine Learning Research* **6**: 1889-1918.

Freund, Y. and Schapire, R. E. (1997). "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of Computer and System Sciences* **55**: 119-139.

Friedlander, M. R., Mackowiak, S. D., Li, N., Chen, W. and Rajewsky, N. (2012). "miRDeep2 accurately identifies known and hundreds of novel microRNA genes in seven animal clades." *Nucleic Acids Res* **40**(1): 37-52.

Gardner, T. S., di Bernardo, D., Lorenz, D. and Collins, J. J. (2003). "Inferring genetic networks and identifying compound mode of action via expression profiling." *Science* **301**(5629): 102-105.

Gomez, A. and Ingelman-Sundberg, M. (2009). "Epigenetic and microRNA-dependent control of cytochrome P450 expression: a gap between DNA and protein." *Pharmacogenomics* **10**(7): 1067-1076.

Gottesman, S. (2005). "Micros for microbes: non-coding regulatory RNAs in bacteria." *Trends Genet* **21**(7): 399-404.

Greenfield, A., Madar, A., Ostrer, H. and Bonneau, R. (2010). "DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models." *PLoS One* **5**(10): e13397.

Griesmer, S. J., Cervantes-Cervantes, M., Song, Y. and Wang, J. T. (2011). "In silico prediction of noncoding RNAs using supervised learning and feature ranking methods." *International Journal of Bioinformatics Research and Applications* **7**(4): 355-375.

Hackenberg, M., Rodriguez-Ezpeleta, N. and Aransay, A. M. (2011). "miRanalyzer: an update on the detection and analysis of microRNAs in high-throughput sequencing experiments." *Nucleic Acids Res* **39**(Web Server issue): W132-138.

Haibe-Kains, B., Olsen, C., Djebbari, A., Bontempi, G., Correll, M., Bouton, C. and Quackenbush, J. (2012). "Predictive networks: a flexible, open source, web application for integration and analysis of human gene networks." *Nucleic Acids Res* **40**(Database issue): D866-875.

Harris, R. S. (2007). "Improved pairwise alignment of genomic DNA." Ph.D. Thesis, Pennsylvania State University.

Hendrix, D., Levine, M. and Shi, W. (2010). "miRTRAP, a computational method for the systematic identification of miRNAs from high throughput sequencing data." *Genome Biol* **11**(4): R39.

Hofacker, I. L. (2003). "Vienna RNA secondary structure server." *Nucleic Acids Res* **31**(13): 3429-3431.

Huang, T. H., Fan, B., Rothschild, M. F., Hu, Z. L., Li, K. and Zhao, S. H. (2007). "MiRFinder: an improved approach and software implementation for genome-wide fast microRNA precursor scans." *BMC Bioinformatics* **8**: 341.

Huynh-Thu, V. A., Irrthum, A., Wehenkel, L. and Geurts, P. (2010). "Inferring regulatory networks from expression data using tree-based methods." *PLoS One* **5**(9).

John, B., Enright, A. J., Aravin, A., Tuschl, T., Sander, C. and Marks, D. S. (2004). "Human MicroRNA targets." *PLoS Biol* **2**(11): e363.

Johnston, R. J. and Hobert, O. (2003). "A microRNA controlling left/right neuronal asymmetry in Caenorhabditis elegans." *Nature* **426**(6968): 845-849.

Kellis, M., Patterson, N., Endrizzi, M., Birren, B. and Lander, E. S. (2003). "Sequencing and comparison of yeast species to identify genes and regulatory elements." *Nature* **423**(6937): 241-254.

Kozomara, A. and Griffiths-Jones, S. (2011). "miRBase: integrating microRNA annotation and deep-sequencing data." *Nucleic Acids Res* **39**(Database issue): D152-157.

Lai, E. C., Tomancak, P., Williams, R. W. and Rubin, G. M. (2003). "Computational identification of Drosophila microRNA genes." *Genome Biol* **4**(7): R42.

Langfelder, P. and Horvath, S. (2008). "WGCNA: an R package for weighted correlation network analysis." *BMC Bioinformatics* **9**: 559.

Lee, R. C., Feinbaum, R. L. and Ambros, V. (1993). "The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14." *Cell* **75**(5): 843-854.

Lee, W. P. and Tzou, W. S. (2009). "Computational methods for discovering gene networks from expression data." *Brief Bioinform* **10**(4): 408-423.

Lim, L. P., Glasner, M. E., Yekta, S., Burge, C. B. and Bartel, D. P. (2003). "Vertebrate microRNA genes." *Science* **299**(5612): 1540.

Lindblad-Toh, K., Garber, M., Zuk, O., Lin, M. F., Parker, B. J., Washietl, S., Kheradpour, P., Ernst, J., Jordan, G., Mauceli, E., Ward, L. D., Lowe, C. B., Holloway, A. K., Clamp, M., Gnerre, S., Alfoldi, J., Beal, K., Chang, J., Clawson, H., Cuff, J., Di Palma, F., Fitzgerald, S., Flicek, P., Guttman, M., Hubisz, M. J., Jaffe, D. B., Jungreis, I., Kent, W. J., Kostka, D., Lara, M., Martins, A. L., Massingham, T., Moltke, I., Raney, B. J., Rasmussen, M. D., Robinson, J., Stark, A., Vilella, A. J., Wen, J., Xie, X., Zody, M. C., Baldwin, J., Bloom, T., Chin, C. W., Heiman, D., Nicol, R., Nusbaum, C., Young, S., Wilkinson, J., Worley, K. C., Kovar, C. L., Muzny, D. M., Gibbs, R. A., Cree, A., Dihn, H. H., Fowler, G., Jhangiani, S., Joshi, V., Lee, S., Lewis, L. R., Nazareth, L. V., Okwuonu, G., Santibanez, J., Warren, W. C., Mardis, E. R., Weinstock, G. M., Wilson, R. K., Delehaunty, K., Dooling, D., Fronik, C., Fulton, L., Fulton, B., Graves, T., Minx, P., Sodergren, E., Birney, E., Margulies, E. H., Herrero, J., Green, E. D., Haussler, D., Siepel, A., Goldman, N., Pollard, K. S., Pedersen, J. S., Lander, E. S. and Kellis, M. (2011). "A high-resolution map of human evolutionary constraint using 29 mammals." *Nature* **478**(7370): 476-482.

Lingeman, J. M. and Shasha, D. (2012). *Network inference in molecular biology*. New York: Springer.

Lippert, R. A., Zhao, X., Florea, L., Mobarry, C. and Istrail, S. (2005). "Finding anchors for genomic sequence comparison." *J Comput Biol* **12**(6): 762-776.

Liu, J., Wang, J. T., Hu, J. and Tian, B. (2005). "A method for aligning RNA secondary structures and its application to RNA motif detection." *BMC Bioinformatics* **6**: 89.

Lu, M., Zhang, Q., Deng, M., Miao, J., Guo, Y., Gao, W. and Cui, Q. (2008). "An analysis of human microRNA and disease associations." *PLoS One* **3**(10): e3420.

Mack, G. S. (2007). "MicroRNA gets down to business." *Nat Biotechnol* **25**(6): 631-638.

Macneil, L. T. and Walhout, A. J. (2011). "Gene regulatory networks and the role of robustness and stochasticity in the control of gene expression." *Genome Res* **21**(5): 645-657.

MacNeil, L. T. and Walhout, A. J. (2011). "Gene regulatory networks and the role of robustness and stochasticity in the control of gene expression." *Genome Res* **21**: 645-657.

Madar, A., Greenfield, A., Ostrer, H., Vanden-Eijnden, E. and Bonneau, R. (2009). "The Inferelator 2.0: a scalable framework for reconstruction of dynamic regulatory network models." *Conference proceedings : 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* **2009**: 5448-5451.

Maetschke, S. R., Madhamshettiwar, P. B., Davis, M. J. and Ragan, M. A. (2013). "Supervised, semi-supervised and unsupervised inference of gene regulatory networks." *Brief Bioinform*.

Marbach, D., Roy, S., Ay, F., Meyer, P. E., Candeias, R., Kahveci, T., Bristow, C. A. and Kellis, M. (2012). "Predictive regulatory models in Drosophila melanogaster by integrative inference of transcriptional networks." *Genome Res* **22**(7): 1334-1349.

Marbach, D., Schaffter, T., Mattiussi, C. and Floreano, D. (2009). "Generating realistic in silico gene networks for performance assessment of reverse engineering methods." *Journal of Computational Biology* **16**(2): 229-239.

Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R. and Califano, A. (2006). "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context." *BMC Bioinformatics* **7 Suppl 1**: S7.

Martens-Uzunova, E. S., Jalava, S. E., Dits, N. F., van Leenders, G. J., Moller, S., Trapman, J., Bangma, C. H., Litman, T., Visakorpi, T. and Jenster, G. (2012). "Diagnostic and prognostic signatures from the small non-coding RNA transcriptome in prostate cancer." *Oncogene* **31**(8): 978-991.

Mathelier, A. and Carbone, A. (2010). "MIReNA: finding microRNAs with high accuracy and no learning at genome scale and from deep sequencing data." *Bioinformatics* **26**(18): 2226-2234.

McEntyre, J. and Lipman, D. (2001). "PubMed: bridging the information gap." *CMAJ* **164**(9): 1317-1319.

Meyer, P. E., Lafitte, F. and Bontempi, G. (2008). "minet: A R/Bioconductor package for inferring large transcriptional networks using mutual information." *BMC Bioinformatics* **9**: 461.

Miller, W., Rosenbloom, K., Hardison, R. C., Hou, M., Taylor, J., Raney, B., Burhans, R., King, D. C., Baertsch, R., Blankenberg, D., Kosakovsky Pond, S. L., Nekrutenko, A., Giardine, B., Harris, R. S., Tyekucheva, S., Diekhans, M., Pringle, T. H., Murphy, W. J., Lesk, A., Weinstock, G. M., Lindblad-Toh, K., Gibbs, R. A., Lander, E. S., Siepel, A., Haussler, D. and Kent, W. J. (2007). "28-way vertebrate alignment and conservation track in the UCSC Genome Browser." *Genome Res* **17**(12): 1797-1808.

Mitchell, P. S., Parkin, R. K., Kroh, E. M., Fritz, B. R., Wyman, S. K., Pogosova-Agadjanyan, E. L., Peterson, A., Noteboom, J., O'Briant, K. C., Allen, A., Lin, D. W., Urban, N., Drescher, C. W., Knudsen, B. S., Stirewalt, D. L., Gentleman, R., Vessella, R. L., Nelson, P. S., Martin, D. B. and Tewari, M. (2008). "Circulating microRNAs as stable blood-based markers for cancer detection." *Proceedings of the National Academy of Sciences U S A* **105**(30): 10513-10518.

Mordelet, F. and Vert, J. P. (2008). "SIRENE: supervised inference of regulatory networks." *Bioinformatics* **24**(16): i76-82.

Mukhopadhyay, N. D. and Chatterjee, S. (2007). "Causality and pathway search in microarray time series experiment." *Bioinformatics* **23**(4): 442-449.

Needleman, S. B. and Wunsch, C. D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of Molecular Biology* **48**(3): 443-453.

Ng, K. L. and Mishra, S. K. (2007). "De novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures." *Bioinformatics* **23**(11): 1321-1330.

Notredame, C. (2007). "Recent evolutions of multiple sequence alignment algorithms." *PLoS Comput Biol* **3**(8): e123.

Ostling, P., Leivonen, S. K., Aakula, A., Kohonen, P., Makela, R., Hagman, Z., Edsjo, A., Kangaspeska, S., Edgren, H., Nicorici, D., Bjartell, A., Ceder, Y., Perala, M. and Kallioniemi, O. (2011). "Systematic analysis of microRNAs targeting the androgen receptor in prostate cancer cells." *Cancer Res* **71**(5): 1956-1967.

Pan, Y. Z., Gao, W. and Yu, A. M. (2009). "MicroRNAs regulate CYP3A4 expression via direct and indirect targeting." *Drug Metab Dispos* **37**(10): 2112-2117.

Paten, B., Earl, D., Nguyen, N., Diekhans, M., Zerbino, D. and Haussler, D. (2011). "Cactus: Algorithms for genome multiple sequence alignment." *Genome Res* **21**(9): 1512-1528.

Paten, B., Herrero, J., Beal, K., Fitzgerald, S. and Birney, E. (2008). "Enredo and Pecan: genome-wide mammalian consistency-based multiple alignment with paralogs." *Genome Res* **18**(11): 1814-1828.

Pearson, W. R. (2000). "Flexible sequence similarity searching with the FASTA3 program package." *Methods in Molecular Biology* **132**: 185-219.

Ribas, J., Ni, X., Haffner, M., Wentzel, E. A., Salmasi, A. H., Chowdhury, W. H., Kudrolli, T. A., Yegnasubramanian, S., Luo, J., Rodriguez, R., Mendell, J. T. and Lupold, S. E. (2009). "miR-21: an androgen receptor-regulated microRNA that promotes hormone-dependent and hormone-independent prostate cancer growth." *Cancer Res* **69**(18): 7165-7169.

Sampson, V. B., Rong, N. H., Han, J., Yang, Q., Aris, V., Soteropoulos, P., Petrelli, N. J., Dunn, S. P. and Krueger, L. J. (2007). "MicroRNA let-7a down-regulates MYC and reverts MYC-induced growth in Burkitt lymphoma cells." *Cancer Res* **67**(20): 9762-9770.

Schapire, R. E. (1999). "A brief introduction to boosting." *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*: 1401-1406.

Scholer, N., Langer, C. and Kuchenbauer, F. (2011). "Circulating microRNAs as biomarkers - True Blood?" *Genome Med* **3**(11): 72.

Schwartz, S., Kent, W. J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R. C., Haussler, D. and Miller, W. (2003). "Human-mouse alignments with BLASTZ." *Genome Res* **13**(1): 103-107.

Sewer, A., Paul, N., Landgraf, P., Aravin, A., Pfeffer, S., Brownstein, M. J., Tuschl, T., van Nimwegen, E. and Zavolan, M. (2005). "Identification of clustered microRNAs using an ab initio prediction method." *BMC Bioinformatics* **6**: 267.

Shojaie, A. and Michailidis, G. (2010). "Discovering graphical Granger causality using the truncating lasso penalty." *Bioinformatics* **26**(18): i517-523.

Slater, G. S. and Birney, E. (2005). "Automated generation of heuristics for biological sequence comparison." *BMC Bioinformatics* **6**: 31.

Smith, T. F. and Waterman, M. S. (1981). "Identification of common molecular subsequences." *Journal of Molecular Biology* **147**(1): 195-197.

Spirollari, J., Wang, J. T., Zhang, K., Bellofatto, V., Park, Y. and Shapiro, B. A. (2009). "Predicting consensus structures for RNA alignments via pseudo-energy minimization." *Bioinformatics and Biology Insights* **3**: 51-69.

Stolovitzky, G., Monroe, D. and Califano, A. (2007). "Dialogue on reverse-engineering assessment and methods: the DREAM of high-throughput pathway inference." *Annals of the New York Academy of Sciences* **1115**: 1-22.

Takagi, S., Nakajima, M., Mohri, T. and Yokoi, T. (2008). "Post-transcriptional regulation of human pregnane X receptor by micro-RNA affects the expression of cytochrome P450 3A4." *Journal of Biological Chemistry* **283**(15): 9674-9680.

Tang, Y. F., Zhang, Y., Li, X. Y., Li, C., Tian, W. and Liu, L. (2009). "Expression of miR-31, miR-125b-5p, and miR-326 in the adipogenic differentiation process of adipose-derived stem cells." *OMICS* **13**(4): 331-336.

Tjaden, B., Goodwin, S. S., Opdyke, J. A., Guillier, M., Fu, D. X., Gottesman, S. and Storz, G. (2006). "Target prediction for small, noncoding RNAs in bacteria." *Nucleic Acids Res* **34**(9): 2791-2802.

Tsuchiya, Y., Nakajima, M., Takagi, S., Taniya, T. and Yokoi, T. (2006). "MicroRNA regulates the expression of human cytochrome P450 1B1." *Cancer Res* **66**(18): 9090-9098.

Turki, T. and Roshan, U. (2014). "MaxSSmap: a GPU program for mapping divergent short reads to genomes with the maximum scoring subsequence." *BMC Genomics* **15**: 969.

Wallace, I. M., Blackshields, G. and Higgins, D. G. (2005). "Multiple sequence alignments." *Current Opinion in Structural Biology* **15**(3): 261-266.

Wang, J. T. L. and Wu, X. (2006). "Kernel design for RNA classification using Support Vector Machines." *International Journal of Data Mining and Bioinformatics* **1**(1): 57-76.

Wang, T., Zhang, X., Obijuru, L., Laser, J., Aris, V., Lee, P., Mittal, K., Soteropoulos, P. and Wei, J. J. (2007). "A micro-RNA signature associated with race, tumor size, and target gene activity in human uterine leiomyomas." *Genes Chromosomes Cancer* **46**(4): 336-347.

Wang, W. L., Chatterjee, N., Chittur, S. V., Welsh, J. and Tenniswood, M. P. (2011). "Effects of 1alpha,25 dihydroxyvitamin D3 and testosterone on miRNA and mRNA expression in LNCaP cells." *Molecular Cancer* **10**: 58.

Watahiki, A., Wang, Y., Morris, J., Dennis, K., O'Dwyer, H. M., Gleave, M., Gout, P. W. and Wang, Y. (2011). "MicroRNAs associated with metastatic prostate cancer." *PLoS One* **6**(9): e24950.

Xu, C. F., Yu, C. H. and Li, Y. M. (2009). "Regulation of hepatic microRNA expression in response to ischemic preconditioning following ischemia/reperfusion injury in mice." *OMICS* **13**(6): 513-520.

Xue, C., Li, F., He, T., Liu, G. P., Li, Y. and Zhang, X. (2005). "Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine." *BMC Bioinformatics* **6**: 310.

Yu, J., Smith, V. A., Wang, P. P., Hartemink, A. J. and Jarvis, E. D. (2004). "Advances to Bayesian network inference for generating causal networks from observational biological data." *Bioinformatics* **20**(18): 3594-3603.

Zhao, D., Wang, Y., Luo, D., Shi, X., Wang, L., Xu, D., Yu, J. and Liang, Y. (2010). "PMirP: a pre-microRNA prediction method based on structure-sequence hybrid features." *Artificial Intelligence in Medicine* **49**(2): 127-132.

Zheng, Y., Hsu, W., Lee, M. L. and Wong, L. S. (2006). "Exploring Essential Attributes for Detecting MicroRNA Precursors from Background Sequences." *Lecture Notes in Computer Science, Springer* **4316**: 15.

Zhong, L., Wang, J. T. L., Wen, D. and Shapiro, B. A. (2012). "Pre-miRNA classification via combinatorial feature mining and boosting." *In Proceedings of the International Conference on Bioinformatics and Biomedicine*: 369-372.

Zhu, H., Wu, H., Liu, X., Evans, B. R., Medina, D. J., Liu, C. G. and Yang, J. M. (2008). "Role of MicroRNA miR-27a and miR-451 in the regulation of MDR1/P-glycoprotein expression in human cancer cells." *Biochem Pharmacol* **76**(5): 582-588.

Zoppoli, P., Morganella, S. and Ceccarelli, M. (2010). "TimeDelay-ARACNE: Reverse engineering of gene networks from time-course data by an information theoretic approach." *BMC Bioinformatics* **11**: 154.

Zuker, M. (2003). "Mfold web server for nucleic acid folding and hybridization prediction." *Nucleic Acids Res* **31**(13): 3406-3415.