NON-CODING RNA IDENTIFICATION IN LARGE-SCALE GENOMIC DATA

By

Cheng Yuan

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science - Doctor of Philosophy

2014

ABSTRACT

NON-CODING RNA IDENTIFICATION IN LARGE-SCALE GENOMIC DATA

By

Cheng Yuan

Noncoding RNAs (ncRNAs), which function directly as RNAs without translating into proteins, play diverse and important biological functions. ncRNAs function not only through their primary structures, but also secondary structures, which are defined by interactions between Watson-Crick and wobble base pairs. Common types of ncRNA include microRNA, rRNA, snoRNA, tRNA. Functions of ncRNAs vary among different types. Recent studies suggest the existence of large number of ncRNA genes. Identification of novel and known ncRNAs becomes increasingly important in order to understand their functionalities and the underlying communities.

Next-generation sequencing (NGS) technology sheds lights on more comprehensive and sensitive ncRNA annotation. Lowly transcribed ncRNAs or ncRNAs from rare species with low abundance may be identified via deep sequencing. However, there exist several challenges in ncRNA identification in large-scale genomic data. First, the massive volume of datasets could lead to very long computation time, making existing algorithms infeasible. Second, NGS has relatively high error rate, which could further complicate the problem. Third, high sequence similarity among related ncRNAs could make them difficult to identify, resulting in incorrect output. Fourth, while secondary structures should be adopted for accurate ncRNA identification, they usually incur high computational complexity. In particular, some ncRNAs contain pseudoknot structures, which cannot be effectively modeled by the state-of-the-art approach. As a result, ncRNAs containing pseudoknots are hard to annotate.

In my PhD work, I aimed to tackle the above challenges in ncRNA identification. First, I designed a progressive search pipeline to identify ncRNAs containing pseudoknot structures. The algorithms are more efficient than the state-of-the-art approaches and can be used for large-scale data. Second, I designed a ncRNA classification tool for short reads in NGS data lacking quality reference genomes. The initial homology search phase significantly reduces size of the original input, making the tool feasible for large-scale data. Last, I focused on identifying 16S ribosomal RNAs from NGS data. 16S ribosomal RNAs are very important type of ncRNAs, which can be used for phylogenic study. A set of graph based assembly algorithms were applied to form longer or full-length 16S rRNA contigs. I utilized paired-end information in NGS data, so lowly abundant 16S genes can also be identified. To reduce the complexity of problem and make the tool practical for large-scale data, I designed a list of error correction and graph reduction techniques for graph simplification.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Introduction to noncoding RNAs

Noncoding RNAs (ncRNAs) are funtional RNA modules that are not translated into protein. They function directly as RNA molecules and play diverse and important biological functions. Studies have hint at an RNA world, in which the roles of ncRNA genes are as important as that of protein coding genes [1, 2, 3]. Many types of ncRNAs function through both their sequences and secondary structures, which are defined by interactions between Watson-Crick and wobble base pairs as well as non-canonical Primary sequences of ncRNAs are assembled as a chain of nucleotides, adenine (A), cytosine (C), guanine (G) and uracil (U). Adenine and guanine are purines, which can form hydrogen bond with pyrimidines uracil and cytosine, respectively. Other base interactions, such guanine-uracil interaction, are also possible. Because of these interactions, primary sequence of ncRNAs can fold into two-dimensional structures. The structure is composed from several fundamental elements, such as hairpin loop, interior loop, stem and bifurcation, as demonstrated in Figure 1.1 [4].

Common types of ncRNA include snoRNA, microRNA (miRNA), ribosomal RNA (rRNA) and short interfering RNAs (siRNAs). SnoRNAs are normally between 70 and 250 nt long. There exist a very large number of snoRNAs, in which most function in rRNA modificaiton and some function in rRNA processing. [5, 6, 7]. MicroRNA s are very short RNA molecules, usually 21 to 22 nt long, that functions in RNA silencing and post-transcriptional regulation

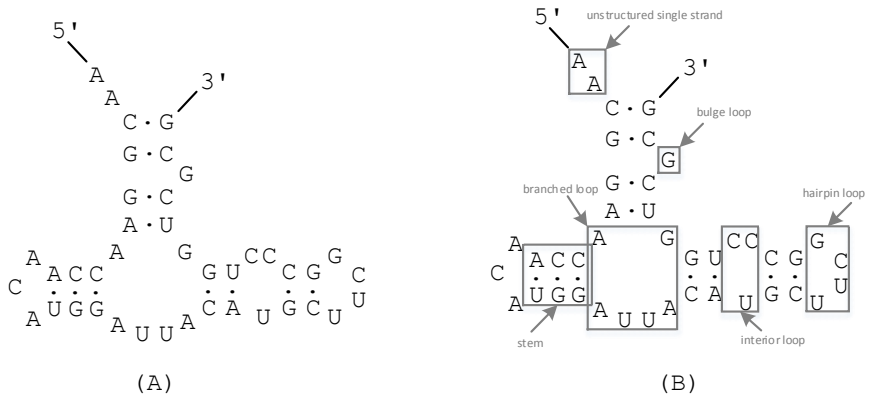

Figure 1.1: An example of the (A) secondary structure and (B) fundamental elements of a hypothetical ncRNA.

of gene expression [8, 9]. Ribosomal RNAs, as predominant material within ribosome, play essential roles in protein synthesis. SiRNAs is a class of double-stranded RNAs, which are involved in in gene silencing in RNA interference (RNAi) pathway [10].

1.2 Related works to ncRNA identification in large-scale genomic data

Recent studies suggest the existence of surprisingly large amount of ncRNA genes in various genomes. In particular, human genomes could possibly contains thousands of ncRNA genes, which play very important role in many biological processes [2]. Thus it becomes increasingly important to identify ncRNA in modern biology. A number of specialized tools are available to identify each type of ncRNAs. For example, tRNAscan-SE [11] is designed to identify tRNAs. snoscan and snoGPS are specialized for snoRNA identification [12]. And miRDeep* [13] can be used to detect miRNAs, while miRPlant [14] aims to find miRNAs only in plants. Each tool may has its specialty in identifying certain type of ncRNAs in certain

domain. There also exist universal tools for ncRNA identification. BLASTN [15] is one of the widely used approaches for nucleotide sequences identification. A BLASTN search performs primary sequence alignments between a query sequences and a library of sequences. A query sequence is identified to be originated from a sequence in library if their alignment score is greater than a threshold. However, only primary sequences of ncRNAs are modeled by BLASTN. NCRNAs having high secondary structure conservation but low primary sequence conservation tend to be missed by BLASTN [16]. The state-of-the-art approach for ncRNA identification approaches are based on stochastic context-free grammar (SCFG). SCFG not only models primary sequences of ncRNAs but also their secondary structures. A detailed explanation can be found in the introduction and method section of Chapter 3. NCRNAs with low sequence conservation but high structural conservation can still be identified. The state-of-the-art implementation of SCFG is covariance model (CM) in Infernal [17]. Each fundamental elements of ncRNAs can be modeled as a state in CM. A query sequence can be optimally aligned to the underlying model using inside-outside algorithms. If the alignment score is above a threshold, the query is then considered to be originated from this model. A number of studies have demonstrated the advantages of incorporating secondary structural information in various types of non-coding RNA homology search [18, 19, 20]. Experiments have been conducted to compare between BLAST and CM-based approaches [18, 20]. The results indicate that BLAST tends to miss ncRNAs with low sequence conservation. The performance of BLAST is even worse with fragmentary query sequences, since the some information is missing, resulting in marginal alignment scores. The latest distribution of Infernal [19], on the other hand, is designed to recover the possible missing bases that could be otherwise form base pairs.

1.3 Challenges in ncRNA identification in large-scale genomic data

1.3.1 Sanger sequencing and NGS technology

Invented in 1977, Sangers method dominated the sequencing market for almost 30 years, and is considered as the "gold standard" for sequencing [21]. The traditional sequencing method relies on DNA synthesis from a DNA template of interest. DNA fragments are polymerized by catalysed enzymic reaction to generate complementary of target DNA sequences. There are some limitation in Sanger sequencing. First, one has to uses gels or polymers as separation media. Second, the number of samples which could be handled in parallel is very limited, making the sequencing process relatively slow. Third, the cost of Sanger sequencing if very high. The limitations triggered the effort to develop more efficient and economical sequencing technologies.

Next generation sequencing (NGS) overcomes the drawbacks of Sanger sequencing. First, with very high throughput, NGS platform can generate massive amount of data in parallel. Whole small genome can now be sequenced in a day [22]. Second, the sequencing cost is greatly reduced. An estimated cost per million sequenced bases using NGS technologies is 1000-fold less than that using traditional chain termination based Sanger sequencing [23]. The entire sequencing process can be done on bench-top instrumentation. Widely used NGS platforms include Roche (454) GS FLX sequencer, Illumina genome analyzer and Applied Biosystems SOLiD sequencer. New platforms such as PacBio and Ion Torrent PGM are also available. The advent of NGS technologies has revolutionized the microbiology study.

NGS technology is utilized by various applications. RNA-sequencing, also called Whole

Transcriptome Shotgun Sequencing (WTSS) [24], is a NGS based application to study RNA. It aims to reveal a snapshot of presenting RNAs in an organism at a given moment. Metagenomics [25] is another application that utilizes NGS techonology and it allows people to study uncultured microorganims in environment samples. Cheaper and faster technologies enable sequencing of uncultured microorganisms directly from their inhabitants. Instead of studying a few carefully cultured species, we are able to screen thousands of species together and discover their relationships. The sequenced data represents a comprehensive and unbiased picture of species diversity in the environmental sample.

1.3.2 NCRNA identification in NGS data

Despite a range of advantages, NGS-based application also has its difficulties and limitations in data analysis. First, volume of NGS data could be very high, making many existing ncRNA identification tools infeasible. For example, RNAv [26] and RNATOP [27] are tools designed for identification of ncRNA with pseudoknot structures. When the size of input data is small, the tools could be very effective. However, if the data volume is high, both tools cannot produce any output in reasonable amount of time [20]. Second, sequences obtained by NGS technologies are usually fragmentary. There is no guarantee that entire sequence of an underlying ncRNA gene is thoroughly sequenced, even with very high coverage during sequencing process. Identifying such ncRNA genes with missing information could be difficult. Sequencing errors could further deteriorate the performance of identification of such genes [28]. Fourth, in particular, metagenomic datasets, which are usually from heterogeneous microbial communities, could contain tens of thousands of species. We do not generally know the ground truth about origin of each fragmentary sequence in the dataset, nor do we know the true microbial composition in the environmental sample, making it

difficult to separate similar ncRNAs genes from highly related species [29]. As a result, effective and efficient algorithms are in need to meet the computational challenges posed by the nature of NGS data.

1.3.3 Pseudoknot structures and challenges in finding ncRNAs containing pseudoknot structures in large-scale genomic data

Pseudoknot is a functionally important structural motif in ncRNA secondary structures [30]. In pseudoknots, bases in loop regions can form base pairs with bases outside the stem loop, as displayed in Figure 1.2. It is already known that pseudoknots play important functions in telomerase RNA, tmRNA, rRNA, some riboswitches, some protein-biding RNAs, Viral ribosomal frameshifting signals, etc [31]. Different research groups [32, 33] have shown that the pseudoknot structure in the telomerase RNA is essential for telomerase activity. Gilley and Blackburn [32] experimentally demonstrated that disruptions of the pseudoknot base pairing within the telomerase RNA from *Tetrahymena thermophila* prevent the stable assembly *in vivo* of an active telomerase. They further concluded that the pseudoknot topology rather than sequence is critical for an active telomerase. Similarly, biologists reported that the pseudoknots in tmRNA are highly important for protein biding, tmRNA maturation, and proper folding of the tRNA-like domain [34].

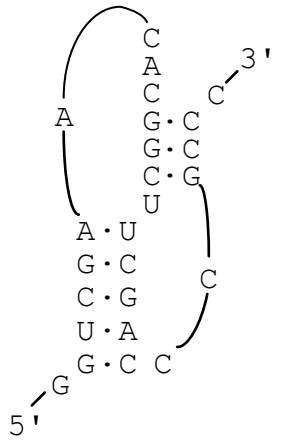

Figure 1.2: An example of a psuedoknot-containing ncRNA

The state-of-the-art approach for ncRNA identification is based on SCFGs. However, SCFGs are not able to model pseudoknot. Thus, the implementations of SCFG by Infernal neglect pseudoknots in the structures. As a result, Infernal could misclassify sequences as members of families containing pseudoknots. In addition, Infernal [19] has high computational cost, limiting its usage in large-scale data sets, such as those generated by the next-generation sequencing technologies. Other approaches include as RNAv [27] and RNATOPS [26]. RNATOPS designs a graph model for RNA pseudoknots and solves the structure sequence alignment by graph optimization.RNAv is a profile based RNA secondary structure variation search program that detects distant ncRNA structural homologs, which might be missed by RNATOPS. Because of the high algorithm complexity, both tools are extremely computationally expensive and are not feasible for large-scale data.

1.4 Contributions

During my Ph.D. study, I tried to tackle a few challenges in ncRNA identification in large-scale genomic data. First, I designed and implemented an efficient progressive search pipeline that can model and search for ncRNAs including pseudoknots. Using the pipeline, one could examine a query to detect each component of a model trained from a family of psuedoknot-containing ncRNAs. Second, I designed and implemented a computational pipeline to identify small ncRNAs in RNA-Seq data. An SCFG-based homology search was applied to significantly reduced, making the pipeline very efficient in processing large dataset. Third, I designed and implemented a set of graph-based algorithms to recover 16s rRNA genes in metagenomic data. I designed a set of graph based algorithms with a list of graph reduction techniques. By utilizing paired-end information, one can separate similar genes from highly related species. Genes with very low abundance can also be recovered.

Chapter 2

Efficient known ncRNA search including pseudoknots

2.1 Background

Noncoding RNAs (ncRNAs), which function directly as RNAs without translating into proteins, play diverse and important biological functions [35]. Many types of ncRNAs function through both their sequences and secondary structures, which are defined by interactions between Watson-Crick and wobble base pairs. Pseudoknot is a functionally important structural motif in ncRNA secondary structures. In pseudoknots, bases in loop regions can form base pairs with bases outside the stem loop. In a graphical representation where arcs connect base pairs, pseudoknot-free secondary structures only contain parallel or nested base pairs while pseudoknot structures allow "crossing" base pairs, shown by an example in Figure 2.1.A.

Because the functions of ncRNAs are determined by both the sequence and structure, successful ncRNA homology search tools must consider both sequence and structural conservations. Existing ncRNA search tools can be divided into two categories. One is commonly referred to "known ncRNA search", which aims to detecting homologs of ncRNAs with annotated secondary structures. The second category includes tools for identifying novel ncRNA genes. This work belongs to the first category and focuses on ncRNAs containing

pseudoknots.

For pseudoknot free ncRNAs, the state-of-the-art search method is based on stochastic context-free grammars (SCFGs), which can accurately model the evolutionary changes of both the sequences and structures of a group of homologous ncRNAs. Commonly used general and specialized known ncRNA search tools such as Infernal [19], RSEARCH [36], and tRNAScan-SE [37] are all based on SCFG. In conjunction with the ncRNA family database Rfam, Infernal has been successfully applied to classify query sequences into different types of ncRNA. However, SCFGs are not able to model pseudoknot. Thus, the implementations of SCFG by Infernal neglect pseudoknots in the structures. For example, although RF00023 (tmRNA) has four pseudoknots, its SCFG only models the knot-free structures, shown in Figure 2.1.B. As a result, Infernal could misclassify sequences as members of families containing pseudoknots. In addition, Infernal has high computational cost, limiting its usage in large-scale data sets, such as those generated by the next-generation sequencing technologies.

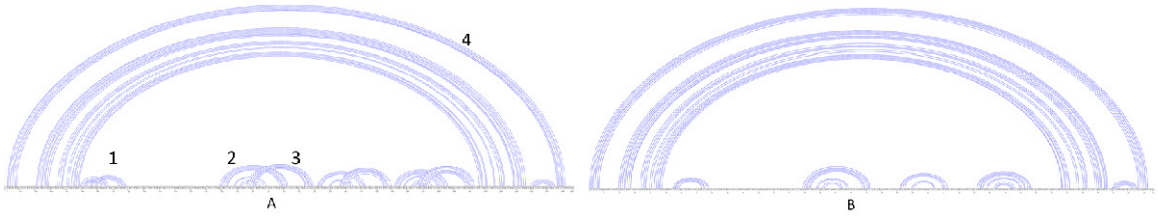

Figure 2.1: Consensus secondary structure of tmRNA and the secondary structure described by SCFG (pseudoknots missing). A. Consensus secondary structure of RF00023 (tmRNA) in Rfam. Stacking base pairs in 1 are parallel to base pairs in 2 and 3. 1, 2, and 3 are nested in 4. 2 and 3 form a pseudoknot. B. Secondary structure described by SCFG (pseudoknots missing).

More complicated grammars such as context-sensitive Grammars (CSGs) [4] exist to faithfully model pseudoknots. However, the computational cost of the parsing algorithms of a CSG is even higher than using a CFG [4]. Besides CSGs, other grammars such as parallel communicating grammar systems [38], RNA pseudoknot grammars [39], tree adjoining

grammars (TAGs) [40, 41], and multiple context-free grammars [42] have been proposed to model pseudoknot structures. These work described the grammars and associated parsing algorithms. However, they have not been widely used in pseudoknot search in large-scale databases. First, although the parsing algorithms are polynomial, their cubic or even higher time or memory complexity [42] limits their large-scale applications. Second, these methods were designed for and tested on secondary structure derivation rather than homology search. In order to conduct large-scale homology search, local parsing algorithms are needed. As there are no source codes or executable implementations of these grammars, it is not clear whether they can be automatically applied to known ncRNA search including pseudoknots.

In this work, we design an efficient pseudoknot search algorithm for all types of pseudoknots. Our method is based on a set of carefully chosen *simple sub-structures* (or *sub-structures* for short), which do not contain pseudoknots or bifurcations. The time complexity of the parsing and probability computation algorithms for an SCFG including the CYK, the inside, and the outside algorithm will be significantly reduced when the secondary structure does not contain any bifurcation [4, 43]. Thus, these simple sub-structures can be searched efficiently using existing implementations of SCFGs. For multiple sub-structures extracted from one ncRNA family, we choose a set of sub-structures according to their sizes and false positive (FP) rates in order to maximize the search performance. These chosen sub-structures will be used in a progressive search. Our experimental results show that our tool competes favorably with other pseudoknot search methods.

2.2 Related Work

Brown and Wilson [44] proposed an RNA pseudoknot search method using intersections of SCFGs. Both Brown's method and our approach try to decompose pseudoknot into knot-free

structures for SCFG modeling. There are two major differences. First, our sub-structures are not only knot-free, but also bifurcation free, which enables faster search. Second, while Brown and Wilson's method focused on the model construction and parsing algorithm, we focus on choosing an optimal set of sub-structures to optimize the search performance. The model construction and the parsing algorithms can be conveniently implemented using Infernal, which has gone through extensive testing.

Structural motifs similar to sub-structures have been used as filters to speed up Infernal. FastR [45] relies on stem-loops ((k, w)-stack) that do not contain bulge or interior loops to search for ncRNAs. Weinberg et al. [46] use more flexible structural motifs based on sub-CMs and profile HMMs for ncRNA classification. Smith [43] used a decision tree to organize partial SCFG models for fast ncRNA search. Currently, these filters are only designed and tested for speeding up SCFG search.

Available pseudoknot search tools include RNAv [27] and RNATOPS [26]. RNATOPS designs a graph model for RNA pseudoknots and solves the structure sequence alignment by graph optimization.RNAv is a profile based RNA secondary structure variation search program that detects distant ncRNA structural homologs, which might be missed by RNATOPS.

The chain filter designed by Zhang et al. [45] consists of a collection of short conserved words in an ncRNA family. In our work, we use a collection of simple sub-structures for pseudoknot search. Similar to Zhang et al.'s work, we find that using a collection of simple structures can achieve a good tradeoff between sensitivity and false positive rate during search.

2.3 Approach

There are two components in the method. The first component is the design of a set of sub-structures to represent an ncRNA family. The second component is a progressive search strategy using the designed sub-structures. Different regions of an ncRNA sequence have different sequence and structural conservations. Well-conserved structural and sequence motifs tend to yield better search performance than poorly conserved motifs. Our approach sorts sub-structures extracted from different regions according to their lengths and predicted FP rates in order to choose a set of sub-structures with the optimal search performance.

For a chosen set of sub-structures, we conduct a progressive search according to a pre-determined order. During the progressive search, one sub-structure is only applied to regions containing matches to all previous sub-structures. A sequence is classified into the pseudo-knot family if and only if 1) it passes the score thresholds of all the chosen sub-structures; 2) the position relationship between matched substrings is consistent with the relationship between the sub-structures. Thus the false positive rate of the chosen set of sub-structures is bounded by the product of the false positive rates of all component sub-structures. The pipeline of the approach is illustrated in Figure 2.2

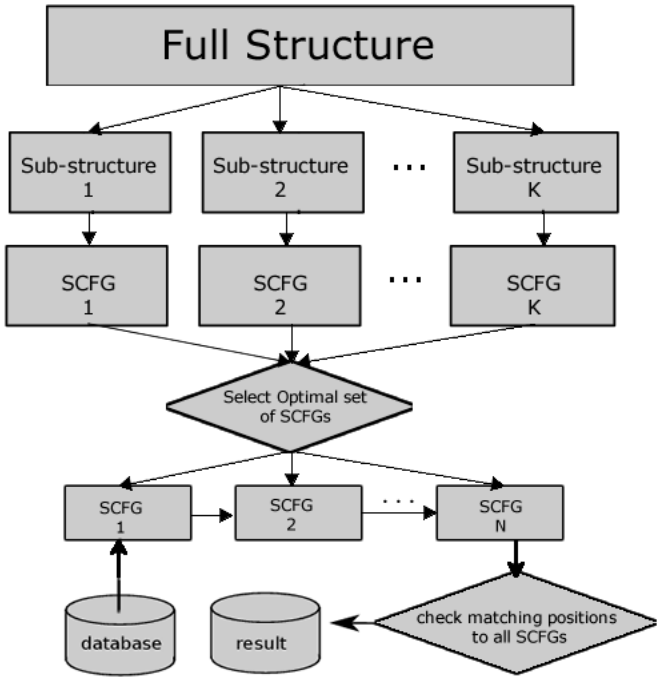

Figure 2.2: The pipeline of the SCFG construction and the progressive search.

2.3.1 Sub-structure derivation

In order to use SCFG-based models for pseudoknot search, we decompose a pseudoknot structure into simple sub-structures. Each sub-structure contains at least one *stem*, which includes a set of stacking base pairs allowing short bulge and interior loops. A full secondary structure of an ncRNA family can be decomposed into multiple stems. Combinations of stems define different sub-structures. Figure 2.3 shows all five simple sub-structures derived from the given pseudoknot.

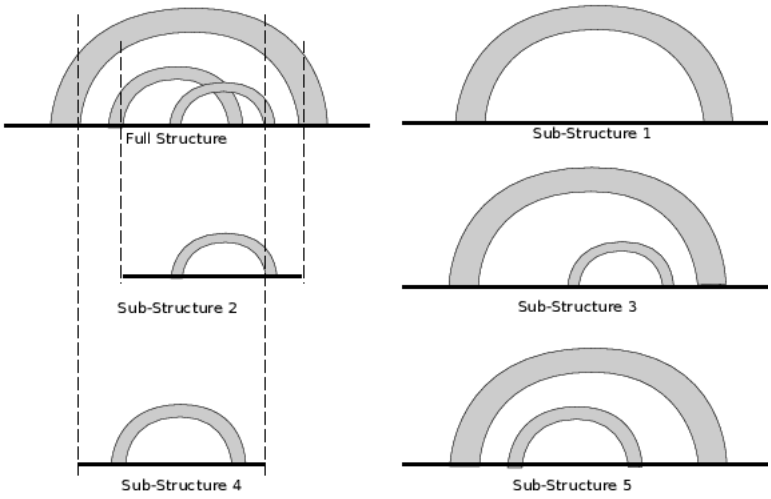

Figure 2.3: Five candidate sub-structures can be constructed from three stems in a pseudoknot structure. Each arc represents a stem containing nested base pairs and possible internal/bulge loops. Single-stranded regions are represented using solid lines.

We describe a method to systematically extract all simple sub-structures from a pseudoknot. In the first step, all stems are extracted and sorted in increasing order of their starting positions (i.e. 5' end of the outmost base pair in the stem). Second, we build a bit table R of size N by N for N stems extracted from the first step. For each cell $R[i, j]$, if stem i and stem j are nested, $R[i, j] = 1$; otherwise, $R[i, j] = 0$. Table R provides us information about whether given stems can form one sub-structure. Given the stem set and their relationship table R, we use pseudocode in **Algorithm 1** to extract all simple sub-structures. In the pseudocode, H^x is the set of sub-structures containing x stems. Thus, the union of H^x for $x = 1$ to N consists of all simple sub-structures for a given secondary structure. The number of sub-structures depends on the number of nested stems. Suppose the average number of nested stems inside a stem is n. The total number of sub-structures is $O(N + N2^n)$.

Algorithm 1 only outputs the combination of stems. For each stem (or stem set) in a sub-structure, we add loop and flanking regions using the following three rules. Let the 5'

Algorithm 1 ExtractSubstructures **Input:** a secondary structure containing pseudoknots **Output:** all simple sub-structures

```
for each stem i = 1 to N do
   /* h: a sub-structure containing a set of stems */
   h = {i}
   H^1 = H^1 ⋃ {h}
end for
for L = 2 to N do
   H^L = ∅
   for each sub-structure h ∈ H^(L-1) do
      for each stem i ∉ h do
         /* h[i] is the ith stem in a sub-structure h */
         if R[h[1],i] and R[h[2],i] ... and R[h[L-1], i] then
            /* construct a new sub-structure h' */
            h' = h ⋃ {i}
            H^L = H^L ⋃ {h'}
         end if
      end for
   end for
end for
output all sub-structures H = H^1 ⋃ H^2 ⋃ ... ⋃ H^N
```

and 3' ends of the outmost base pair in a sub-structure be I_5 and I_3, respectively. Thus, $I_5 < I_3$.

- Rule 1: Add all single-stranded regions including bulge and internal loops between I_5 and I_3.
- Rule 2: Except the base pairs inside the chosen stems in a sub-structure, all other base pairs will be treated as single-stranded regions.
- Rule 3: Extend the flanking single-stranded regions to the left of I_5 and to the right of I_3 until the first base pair in other sub-structures.

2.3.2 Search performance of different sub-structures

Each sub-structure can be conveniently modeled by an SCFG. As different sub-structures are derived from regions with different sequence and structural conservations, their corresponding SCFGs have different performance in database search. In this section, we use an example to illustrate this. We built SCFGs for eight sub-structures derived from RF00373 (Ribonuclease P) and evaluated the sensitivity, FP rates, and running time of the eight SCFGs when applying them to a to a 22.5M Maize chromosome (data is described in "Experimental results"). The sensitivity and FP rates of different sub-structures from the same family can be compared using true positive (TP) hits and FP hits respectively, because the condition positive and condition negative sets are the same for all sub-structures derived from the same family. For any SCFG $\mathcal{M}_i$, let the set of matched sequences be $Hit(\mathcal{M}_i)$. Let the set of true pseudoknot sequences be S, which are the sequences in seed families containing pseudoknots in Rfam. The number of true positive and FP matches of a sub-SCFG is $|Hit(\mathcal{M}_i) \bigcap S|$ and $|Hit(\mathcal{M}_i) \setminus S|$, respectively. We summarized the TP hits and FP matches of eight SCFGs under different score thresholds in Figure 2.4. In addition, the search times are included for the score thresholds corresponding to the highest sensitivity. It is clear that different SCFGs have highly search performance. During a progressive search using a series of sub-structures, the number of matches of the preceding sub-structure determines the search space of the current sub-structure. Thus, the total search time depends on both the FP hits and the model running time, which is heavily affected by the model length. In order to maximize the search efficiency, it is important to sort all candidate sub-structures according to their FP rates. When the FP rates of two or more sub-structures are similar (same order), we prefer shorter models because they incur less search times.

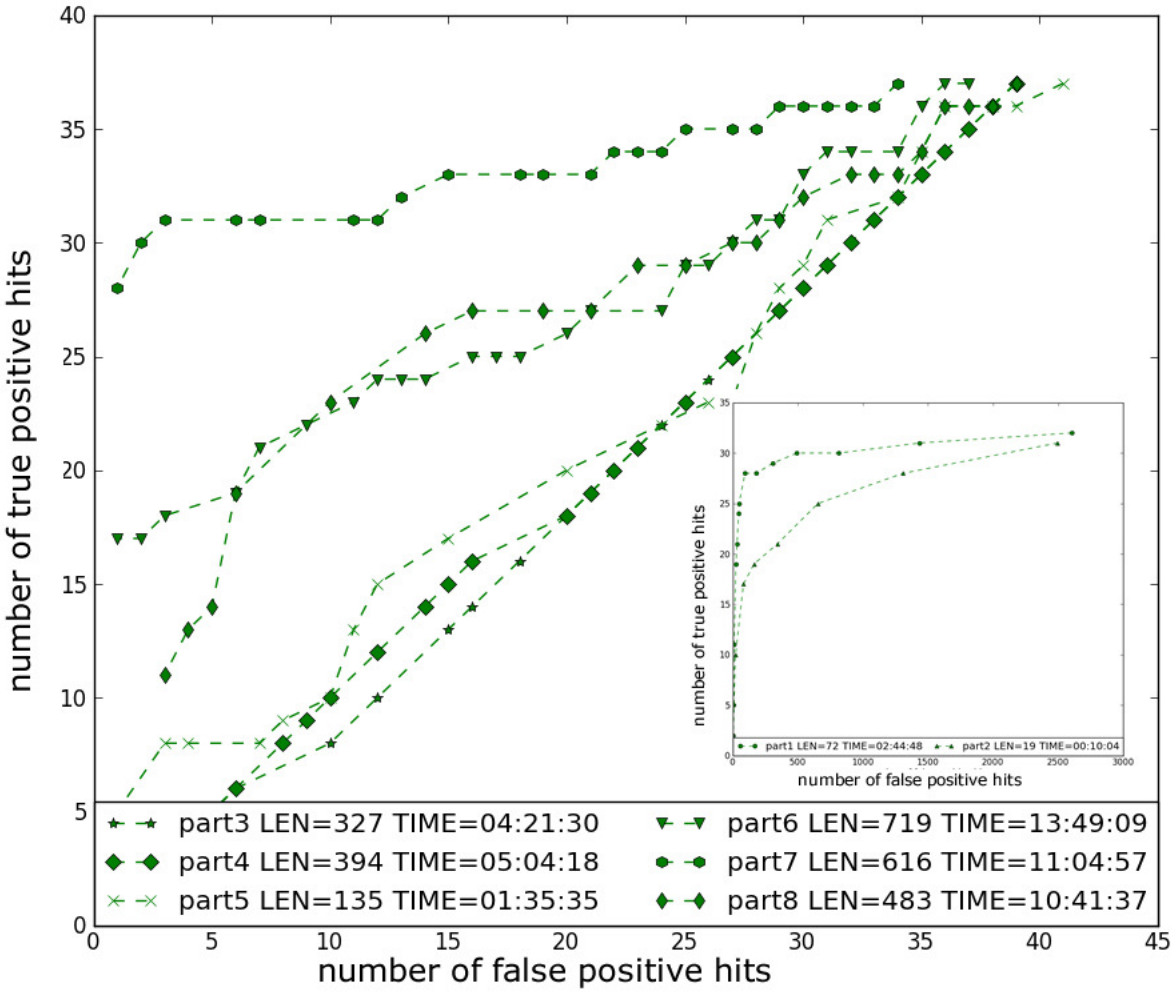

Figure 2.4: Number of TP hits and FP matches of each sub-structure under different score thresholds. For each sub-structure, the length and the search time corresponding to the highest sensitivity is listed. Time format is hr:min:sec. Due to highly different number of FP hits, two sub-structures are plotted in the embedded figure.

2.3.2.1 Sort sub-structures according to their E-values

There are two methods to calculate the FP rates of sub-structures. Theoretically, by assuming a background model for random sequences and applying the CYK algorithm [4], we can directly calculate the probability that a random sequence matches an SCFG model.

Empirically, we can apply the SCFGs to a large annotated sequence database and record the number of FP matches. However, as it is more important to compare the FP rates of different sub-structures than knowing their exact values, it is not necessary to directly calculate FP rates. By assuming that the SCFG alignment scores for random sequences follow an exponential distribution, as implemented by Infernal, we can use E-values of the designed score cutoffs to sort all sub-structures.

For an alignment score and a database size, an E-value indicates how many random hits a user can expect to see with the same or better score in a random sequence database of similar size. Thus, E-value indicates FP hits when it can be computed accurately. Currently, we are using the E-value calculation method provided by Infernal. Although the assumed score distribution is not accurate, we found that the estimated E-values allow us to compare FP rates of different sub-structures with high accuracy. In order to estimate E-value, Infernal generates a set of N random sequences whose GC content depends on the covariance model. These N random sequences then are aligned against the model. In this process, all searching result with score > 0 will be considered as hits. Scores of the top X hits are assumed to follow an exponential distribution with two parameters, μ and λ. The maximum likelihood approach is then taken to fit scores of hits into an exponential distribution.

$$E = db * e^{-\lambda(score-\mu)}$$

where db is adjusted database size and is defined as

$$db = \frac{db_{target}}{dbsize_{random}}(randhit + 0.5).$$

In the E-value computation, μ and λ are parameters trained in Infernal. sc is the score for which one needs to calculate E-value. db_{target} is the size of target database. db_{random} is the number of random sequences generated for curve fitting. At last, $randhit$ is the number of random sequences found by the covariance model. We can directly obtain μ and λ from each calibrated covariance model, which is built for a sub-structure. With these two parameters available, we can use the above equation to compute E-values for given scores.

Our experiments show that although the change of E-values does not scale with the change of the FP rates, the order of E-values is highly consistent to the order of FP rates for all 71 families we tested. Only for SCFGs with similarly small FP rates, their E-values cannot accurately reflect their order. Table 2.1 presents an example. It is worth noting that we also considered to use the average entropy to sort the sub-structures. However, our experiments show that there is no systematic relationship between entropy-based measurements and the FP rates of sub-structures.

sub-structure	E-value	FP hits	sub-structure	E-value	FP hits
RF00373_part2	1.71e+03	4894	RF00373_part1	7.33e+02	2606
RF00373_part5	7.30e-02	41	RF00373_part3	3.58e-02	39
RF00373_part4	3.18e-06	39	RF00373_part6	5.29e-09	37
RF00373_part8	4.52e-09	39	RF00373_part7	3.40e-15	34

Table 2.1: The order of E-values is highly consistent to the order of number of the FP hits.

2.3.2.2 Choose sub-structures for progressive search

During a progressive search based on multiple sub-structures, the final sensitivity is bounded by the lowest sensitivity of all sub-structures. The final search time and FP rates heavily

depend on the order of applying these sub-structures. Let the final array of sub-structures be $\mathcal{SUB}$=($\mathcal{H}_1, \ldots, \mathcal{H}_i, \ldots, \mathcal{H}_n$), where $\mathcal{H}_i$ will be applied before $\mathcal{H}_j$ if $i < j$. Let the size of the original database be L. For a sub-structure $\mathcal{H}_i$, let t_i and fp_i be its search time per hit and FP rate, respectively. The final FP rate is bounded by $\prod_{i=1}^{n} fp_i$. The final search time is roughly $T = L\sum_{i=1}^{n} t_i(\prod_{j=1}^{i-1} fp_j)$, where $L\prod_{j=1}^{i-1} fp_j$ is roughly the search space for the sub-structure $\mathcal{H}_i$. Minimizing T requires the accurate computation of t_i or quantification of the relationship between t_i and fp_i, which is not known as a priori. Although Infernal provides estimated running time, it can be quite different from the true running time. According to the equations, it is clear that we should apply short sub-structures with small FP rates before long sub-structures with high FP rates. Thus we develop a greedy algorithm to generate a set of sub-structures for progressive search based on our empirical observations.

We split sub-structures into *short* group and *long* group, which contain short and long sub-structures respectively. For each group of sub-structures, we sort the sub-structures according to their E-values and apply a greedy algorithm to choose a set of sub-structures for search. The main steps of the greedy algorithm are outlined below, starting from the short group:

1. In each iteration, choose the sub-structure with the smallest E-value. Remove it and append it to the final sub-structure list $\mathcal{SUB}$.
2. Remove any remaining sub-structure in both groups that only contains stems in this sub-structure.
3. Repeat the first step until all stems are covered by one chosen sub-structure or the E-values of all remaining sub-structures are bigger than a pre-determined cutoff (default is 1).

If $\mathcal{SUB}$ has not included all stems, we apply the same process to the long group and append the chosen sub-structures to $\mathcal{SUB}$. We require all stems covered by the chosen sub-structures in order to ensure the representation of the annotated pseudoknot structure. It is possible that this constraint will exclude homologous ncRNAs that lack annotated stem loop structures. Currently, we use size 150 as the threshold to divide sub-structures into the short and the long group.

2.3.3 Implementation

For each sub-structure, we train an SCFG-based model based on the corresponding alignment in the training data using Infernal. Let the SCFGs trained from n sub-structures of an ncRNA family $\mathcal{SUB} = (\mathcal{H}_1, \ldots, \mathcal{H}_i, \ldots, \mathcal{H}_n)$ be $\Pi = (\mathcal{M}_1, \ldots, \mathcal{M}_i, \ldots, \mathcal{M}_n)$, where $\mathcal{M}_i$ represents a single SCFG. A sequence can be classified into the corresponding family if the following conditions are satisfied. First, the sequence contains matches to all designed SCFGs in Π. SCFG match will be defined in the following text. Second, for every pair of strings that match two SCFGs, their position relationship must be consistent with the annotated relationship between two SCFGs in the underlying ncRNA family. There are three types of position relationship between two sub-structures: parallel, nested, and cross-over. Cross-over indicates existence of pseudoknots.

We determine SCFG match using score thresholds. For all sequences in the training set, its alignment score with a given SCFG is computed. The minimum score of all the seed sequences is used as the score threshold. This score cutoff is similar to the NC (trusted cutoff) bit score thresholds used in HMMER [47] or Infernal. When the training data contains a good representation of the family member sequences, the computed score threshold can ensure a high sensitivity during homology search. If the training set only contains close homologs of

this ncRNA family, the designed cutoff may be too high for remotely related homologs.

2.4 Experimental results

In order to test the performance of our tool for pseudoknot search in sequence databases, we conducted two experiments. First, we examined the automatically classified pseudoknot sequences in Rfam. Second, we applied it to part of the Maize genome. On the same data set, we compared our tool with RNAv, RNATOPS, and Infernal.

2.4.1 Pseudoknot sequences in Rfam

Because CFG cannot model pseudoknots, the implementations of Stochastic CFG (SCFG), covariance models (CMs) in Rfam neglect pseudoknots in the structures. As a result, tools that use SCFG for ncRNA search such as Infernal could misclassify sequences as members of pseudoknot families. Each Rfam family contains a seed sequence set and a full sequence set. While the seed sequence set contains manually validated homologous sequences, the full sets are automatically produced using SCFG-based search against RFAMSEQ database [48]. Thus, some of the sequences in the full set may not contain pseudoknot structures that are annotated in the seed sequences. We examined the full member set of the 71 ncRNA families containing pseudoknots in Rfam using our tool. Many families contain dozens of sequences that lack the annotated pseudoknot structures. For all those sequences that cannot be matched by our tool, we also utilized the Infernal alignments and a RNA stem finding tool RNAmotif [49] to double check whether the base pairs in pseudoknot structures are missing. The SCFG alignments output by Infernal contains annotations of all base pairs that do not form pseudoknots. By comparing the annotated base pairs and the consensus secondary structure of the seed alignments, we can extract the regions that should form pseudoknots.

Then, we applied RNAmotif to output all stems of size at least two in the chosen regions. Failing to output any stems validated our findings that these sequences do not have the annotated pseudoknots. The results are summarized in Table 2.2. Although homologous ncRNAs may not share the same set of stems, simply ignoring pseudoknots without knowing their impacts on the function can introduce a large number of false members. In particular, it was already experimentally shown that pseudoknot structures are vital to the functions of some types of ncRNAs [32, 33, 34]. For these well-studied pseudoknot structures, it is important to include them during homology search.

ID	seqs without knots/ num of seqs	ID	seqs without knots/ num of seqs	ID	seqs without knots/ num of seqs	ID	seqs without knots/ num of seqs
RF00009	37/500	RF00010	3/3864	RF00011	26/460	RF00023	53/2871
RF00024	56/233	RF00028	2587/39045	RF00030	47/470	RF00041	2/151
RF00140	81/524	RF00176	37/64	RF00216	25/126	RF00233	22/76
RF00259	78/124	RF00261	43/78	RF00499	1/16	RF00523	2/5177
RF00622	1/94	RF01050	3/60	RF01072	21/271	RF01073	1/7006
RF01086	15/1093	RF01087	1/31	RF01089	4/25	RF01096	2/45

Table 2.2: Sequences that do not contain annotated pseudoknots and thus may not be real members.

2.4.2 Data set preparation

We created a simulated data set based on a contiguous 22-Mb region of the Maize Genome [50]. The annotation of the 22-Mb region does not contain any hit to the 71 pseudoknot families in Rfam. In order to evaluate the sensitivity of pseudoknot search tools, we randomly chose 1,586 out of 26,704 seed sequences from 71 pseudoknot families and inserted them in the 22-Mb region. The remaining seed sequences are used as the training data. In

order to examine the FP rate of SCFG-based tools, we also created 1,586 sequences without pseudoknots. Specifically, for each of the 1,586 seed sequences, we altered the bases to disrupt the base pairs that can form pseudoknots. Similarly RNAmotif is applied again to ensure these sequences lose the annotated pseudoknot structure. These modified 1,586 sequences and the original 22-Mb region of the Maize Genome constitute the negative training data. Any hit to them is an FP hit. Note that by changing the bases, the modified sequences might share lower sequence similarity to the trained model and thus pose an easier case for all tools. Even so, our experimental results still show that different tools exhibit highly difference performance on this data set. Thus, we feel this data set is a reasonable test set.

There are two major advantages of using this simulated data set for testing pseudoknot search tools. First, as the 22-Mb region of the Maize genome does not harbor any reported ncRNA that contains pseudoknots, we can measure the empirical FP rates of pseudoknot search tools with higher reliability than using simulated sequences, which are usually generated using a simple i.i.d. model or low-order Markov model. In particular, the Maize genome contains a high percentage of repeats and low-complexity regions, which could not be simply simulated and can pose a challenge for ncRNA search as warned by the Rfam website (http://rfam.sanger.ac.uk/). Second, using thousands of seed members of the pseudoknot families provides us adequate test data for evaluating the sensitivity.

Besides using the seed sequences of Rfam, we also considered another pseudoknot sequence database Pseudobase [51]. This database contains 304 RNA sequences with pseudoknot structures. A majority of them are sub-strings of Rfam seed sequences. Thus, we choose to use Rfam seed sequences as the true label.

2.4.3 Results and comparisons

In order to separate the training set and the test set, we removed the sequences that were inserted in the Maize genome from the seed alignments. For the alignments composed of the remaining sequences, we trained the full covariance model and the models for the sub-structures. We used the designed sub-structure sets for pseudoknot search. We evaluated the performance of pseudoknot search tools using three metrics: the sensitivity, FP hits, and running time. For each ncRNA family represented by an SCFG $\mathcal{M}$, let $Hit(\mathcal{M})$ be the set of output sequences by a search tool. Let S be the set of true pseudoknot sequences, which, in this data set, includes seed sequences of each pseudoknot family. The sensitivity is thus defined as:

$$sensitivity = \frac{|Hit(\mathcal{M}) \bigcap S|}{|S|}$$

Any output that does not overlap with true pseudoknot sequences is a false positive hit. The number of FP hits of a search tool on one family is computed as:

$$FP\ hits = |Hit(\mathcal{M}) \setminus S|$$

We report the FP hits instead of the FP rates for two reasons. First, the condition negative set is family specific and thus is the same for all search tools for a given family. Second, the size of the condition negative set is mainly determined by the size of the genome minus the size of all true pseudoknot sequences. For a large genomic sequence, the FP rate becomes very small and cannot reflect the difference between different tools.

On the same dataset, we run RNAv, RNATOPS, and Infernal 1.0.2. Of the three, RNAv and RNATOPS are designed for pseudoknot search. For Infernal and sub-structure, no

hidden Markov model-based filtration was used in order to maximize the sensitivity. Other parameters were set as default for Infernal. We used the default parameters to run RNAv and RNATOPS. All experiments were run on the main cluster of the High Performance Computing Center on campus (http://www.icer.msu.edu/?q=hpcc). Each experiment was allocated four CPU days at most. There are 65 families and 31 families that failed RNAv and RNATOPS, respectively. The search jobs for those families were killed by the cluster after four CPU days. No results were produced. Thus we could not report the results for those families. RNAPTOPS output results for 22 families by the end of the allocated time.

The performance of these four tools is recorded in Table 2.3. The results show that our tool is significantly faster than RNATOPS and RNAv. For a majority of families, the running time is smaller than half an hour. A closer examination reveals that 99% of the running time is attributed to the first sub-structure, which is expected. Of the six families for which RNAv successfully generated outputs, they all have the sensitivity of 1.0, equal to the sensitivity of sub-structure based search. Of the 40 families for which RNATOPS reported results, 14 of them have equal sensitivity to ours. 1 family yields slightly better sensitivity than ours while other 24 families have significantly worse sensitivity. Thus, overall, our search achieves higher sensitivity than RNAv and RNATOPS. In addition, sub-structure based search tool incurs lower FP rate than RNATOPS and RNAv. Table 2.3 shows that RNATOPS yields low FP hits. Of the 40 families, RNATOPS has the same number of FP hits as ours for only one family and significantly more FP hits for the rest. In particular, RNATOPS outputs over 1,000 hits for 9 families.

We compared the sensitivity, FP hits, and running time of Infernal and our tool in Figure 2.5, Figure 2.6, and Figure 2.7 using X-Y scatter plots. As Infernal and our tool generate the same sensitivity or other metrics for some families, we use the bubble plot to visualize

RNA fam ID	sen	FP	time RNAv	sen	FP	time RNA-TOPS	sen	FP	time Sub-structure	sen	FP hits	time INFER-NAL
RF00009							**1.0**	**5**	**01:47:37**	**1.0**	38	26:16:07
RF00010							0.58	**95**	**00:18:47**	**0.97**	318	17:54:31
RF00011							0.84	**25**	**00:06:51**	**0.97**	179	09:09:52
RF00023							0.4	**1**	**00:06:54**	**1.0**	180	13:40:31
RF00024							**0.95**	**24**	**00:06:33**	0.81	86	20:36:42
RF00028							**0.83**	**6**	**22:30:56**	0.72	37	79:05:16
RF00030							0.38	**26**	**02:35:01**	**0.98**	87	83:37:31
RF00041							0.95	**0**	**00:10:37**	**1.0**	64	01:27:52
RF00094							0.88	**0**	**00:09:21**	**1.0**	35	00:54:20
RF00140							0.97	**0**	**01:05:08**	**1.0**	33	01:52:09
RF00165				0.21	4	4days	**1.0**	**0**	**00:22:10**	**1.0**	14	00:32:25
RF00176	**1.0**	58077	19:54:40				**1.0**	**0**	**00:05:48**	**1.0**	21	00:50:54
RF00216							0.87	**0**	**00:03:03**	**1.0**	30	04:42:29
RF00233				0.26	0	4days	0.96	**0**	**00:09:06**	**1.0**	29	00:47:38
RF00259							**1.0**	**0**	**00:05:41**	**1.0**	5	02:09:52
RF00261							**1.0**	**0**	**00:13:53**	**1.0**	20	02:50:11
RF00373							0.92	**27**	**01:35:35**	**0.95**	363	14:15:43
RF00381				0.38	30	4days	**1.0**	**0**	**00:17:10**	**1.0**	15	00:33:42
RF00390				**1.0**	763	4days	**1.0**	**0**	**00:05:21**	**1.0**	6	00:07:35
RF00458							**1.0**	**0**	**00:09:37**	**1.0**	10	02:18:47
RF00499							**1.0**	**0**	**00:09:51**	**1.0**	115	01:33:43
RF00505				0.2	2	4days	**1.0**	**0**	00:32:27	**1.0**	5	**00:29:55**
RF00507				0.41	7	4days	0.95	**0**	**00:34:44**	**1.0**	23	00:52:44
RF00523				0.29	160	4days	0.95	**24**	00:20:31	**1.0**	145	**00:19:24**
RF00622							**1.0**	**0**	**00:05:15**	**1.0**	14	00:42:40
RF01050							**1.0**	**0**	**00:41:32**	**1.0**	13	39:22:21
RF01072				0.52	273	4days	0.96	**0**	**00:08:37**	**1.0**	30	00:10:13
RF01073	**1.0**	196631	13:13:32	0.11	3	4days	**1.0**	**0**	**00:18:36**	**1.0**	13	00:29:04
RF01074				0.5	91	4days	**1.0**	**0**	**00:06:59**	**1.0**	10	00:15:00
RF01075							**1.0**	**0**	**00:07:59**	**1.0**	7	01:00:46
RF01076	**1.0**	139249	16:20:29				**1.0**	**0**	**00:20:36**	**1.0**	5	00:35:33
RF01077							**1.0**	**0**	00:53:32	**1.0**	4	**00:37:23**
RF01078							**1.0**	**0**	**00:12:38**	**1.0**	3	00:26:04
RF01079				**1.0**	333	4days	**1.0**	**0**	**00:07:37**	**1.0**	3	00:16:01
RF01080				0.5	135	4days	**1.0**	**0**	**00:08:04**	**1.0**	110	00:13:41
RF01081				0.67	284	4days	**1.0**	**0**	**00:06:47**	**1.0**	3	00:08:44
RF01082				0.5	2934	4days	**1.0**	**0**	**00:05:47**	**1.0**	4	00:09:13
RF01083				**1.0**	3002	4days	0.67	**1**	**00:04:34**	**1.0**	7	00:07:05
RF01084							**1.0**	**0**	**00:10:46**	**1.0**	8	01:53:25
RF01086							**1.0**	**11**	**05:18:38**	**1.0**	13	05:39:23
RF01087				0.5	3	4days	**1.0**	**0**	**01:19:41**	**1.0**	12	01:37:01
RF01088							**1.0**	**0**	00:39:09	**1.0**	4	**00:37:14**
RF01089				0.33	1	4days	**1.0**	**3**	**01:03:09**	**1.0**	20	01:21:27
RF01090				0.43	4	4days	**1.0**	**0**	**00:23:11**	**1.0**	8	00:36:35
RF01091							**1.0**	**0**	**00:13:06**	**1.0**	4	00:28:51
RF01092	**1.0**	165990	10:58:02	**1.0**	**0**	4days	**1.0**	**0**	**00:17:57**	**1.0**	15	00:30:08
RF01093				0.42	67	4days	**1.0**	**0**	**00:13:59**	**1.0**	23	00:29:56
RF01094							**1.0**	**0**	**00:52:46**	**1.0**	3	01:10:47
RF01095							**1.0**	**0**	**00:10:56**	**1.0**	2	00:27:12
RF01096	**1.0**	166314	16:44:20	0.5	1	4days	**1.0**	**0**	**00:23:04**	**1.0**	2	00:24:45
RF01097				0.25	1	4days	**1.0**	**0**	**00:12:18**	**1.0**	4	00:22:09

Table 2.3: Sensitivity, FP hits, and running time comparison between RNAv, RNATOPS, Infernal, and sub-structure. Bold font is applied to the highest sensitivity, the lowest FP hit, or the shortest running time for each RNA family. The empty cells indicate that the corresponding tools did not generate any output within 4 CPU days.

the number of the same data points. As expected, Infernal is highly sensitive. However, it reported dozens of hits on the pseudoknot-free sequences which we inserted as false positive sequences. For all families, Infernal reported equal or more FP hits than our tool. In addition, it is generally slower than sub-structure-based tool. Out of 71 RNA families, sub-structure-based tool has shorter running time on 66 families. For 14 families, it yields 10x speed up over Infernal.

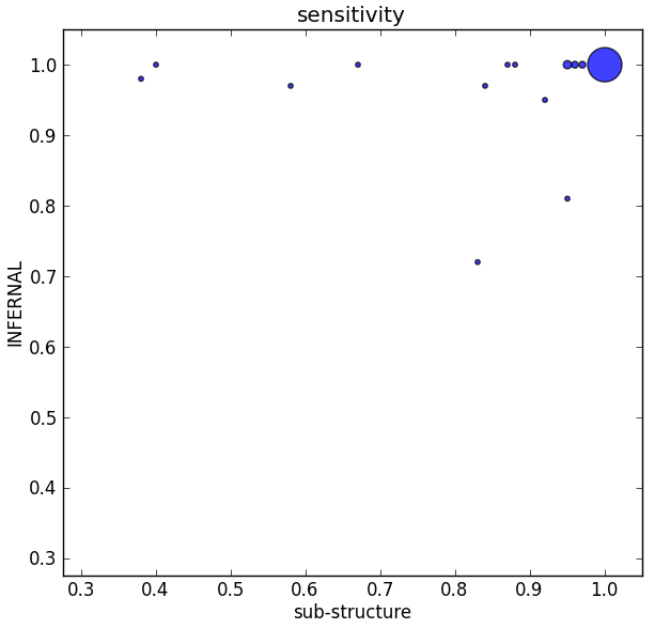

Figure 2.5: Sensitivity comparison on 71 families.

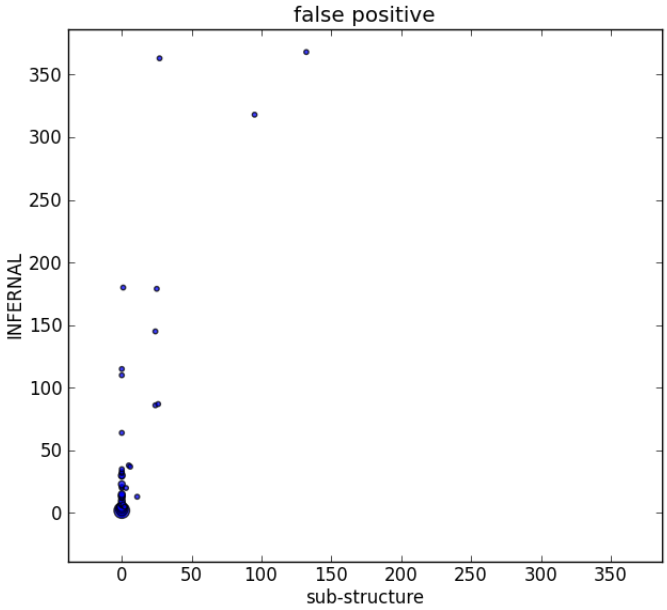

Figure 2.6: Comparison of false positive hits on 71 families.

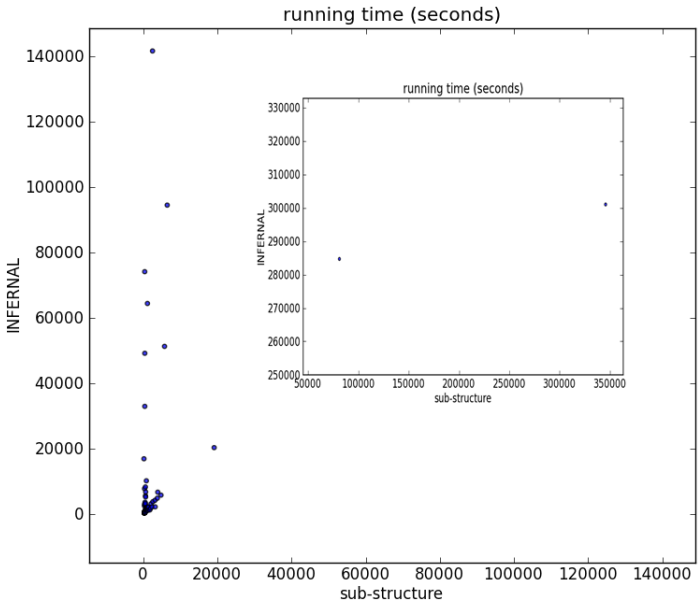

Figure 2.7: Running Time comparison. There are 4 families on which Infernal run much longer than on other families. To keep an appropriate scale, there running times are not displayed on the figure.

There is no significant difference in the sensitivity between Infernal and sub-structure-

based tool when the average sequence length in a family is not too long. Infernal has better sensitivity on longer and more complicated RNA families including RF00010, RF00011, RF00023, and RF00030. The major reason behind our worse sensitivity on the long families is that we use sub-structure that cover every stem. Thus, we only classify sequences that have all characterized stems from the underlying structure. However, some remote homologs may lose base pairs in stems during evolution. Thus while we guarantee to find sequences that have the same structures as the annotated pseudoknots, we can miss some homologs, leading to lower sensitivity for some families.

2.5 Conclusion

Although Infernal is highly sensitive in known ncRNA search, caution must be taken when applying Infernal to ncRNA families containing pseudoknots. In this work, we designed a pseudoknot search method based on a set of carefully chosen sub-structures. These sub-structures do not contain pseudoknots or bifurcations. SCFGs can be conveniently built on them and searched with high efficiency. In order to minimize the overall FP rate and the running time, we sorted sub-structures according to their lengths and their E-values for designed trusted cutoff (NC) bit score thresholds. We designed a greedy algorithm to choose a set of sub-structures and applied the progressive search to minimize search time. Our experimental results showed that our tool competes favorably with RNAv and RNATOPs, both of which have been used for pseudoknot search in large databases. This work provides a complementary pseudoknot search tool to existing SCFG-based knot-free ncRNA search methods.

Currently our tool only reports homologous ncRNAs with the same number of charac-

terized stems as the training data. As a result, some true homologs that have lost one or multiple stems will be ignored. As part of the future work, we plan to incorporate available RNA-seq data for remote homology search.

Chapter 3

RNA-CODE: a noncoding RNA Classification tOol for short reaDs in NGS data lacking rEference genomes

3.1 Introduction

Noncoding RNAs (ncRNAs), which function directly as RNAs without translating into proteins, play diverse and crucial roles in many biochemical processes. For example, tRNAs and rRNAs aid protein synthesis. SnoRNAs guide rRNA modifications. MicroRNAs (miRNAs) regulate gene expression [52]. Short interfering RNAs (siRNAs) involve in gene silencing in RNAi process [53].

In particular, the development of next-generation sequencing (NGS) technologies sheds light on more sensitive and comprehensive ncRNA annotation. Deep sequencing of transcriptomes of various organisms has revealed that a large portion of transcriptomic data cannot be mapped back to annotated protein-coding genes in the reference genome, indicating that those transcripts may contain transcribed ncRNAs [2]. Total RNA-seq and small RNA-seq data generated by numerous transcriptomic sequencing projects are still accumulating rapidly. Identifying different types of ncRNAs and quantifying their expression levels in

different tissues, conditions, and developmental stages have generated new knowledge about functions of ncRNAs. Besides RNA-seq data, ncRNA identification is also important for analyzing metagenomic data, which contain sequenced metagenomes from various environmental samples. For example, 16s rRNA classification [54, 55] and assembly [29, 56] is a fundamental step for studying phylogenies in a sample. NCRNA annotation is, therefore, an important component in post-NGS analysis.

There are two different ncRNA identification problems for NGS data. One focuses on identifying homologs of annotated ncRNAs, such as tRNA, rRNAs, snoRNAs, and many types of miRNAs. Some example applications include comparing expression level changes of let-7 miRNA genes in different developmental stages of *C.* elegans [57], identifying all homologs to annotated miRNAs in the small RNA-seq data of a non-model species [58], and 16s rRNA annotation in metagenomic data [29]. These studies aim to annotate all known ncRNAs or novel members of characterized ncRNA families. The second category focuses on reporting novel ncRNA genes. One possible strategy is to cluster sequences and then apply *de novo* ncRNA gene finding tools such as RNAz[59]. This work belongs to the first category.

The state-of-the-art method for ncRNA homology search is still based on comparative ncRNA identification, which searches for ncRNAs through evidence of evolutionary conservation. As the function of an ncRNA is determined not only by its sequence but also by its secondary structure, which contains interacting base pairs, such as Watson-Crick base pairs and G-U base pairs, successful ncRNA search should take advantage of both sequence and secondary structural similarity. A number of such tools are available such as a general ncRNA search tool Infernal [17] and specialized tools for tRNA [11] and snoRNA [60] etc. However, most existing homology search strategies use complete secondary structure of

annotated ncRNAs and are not optimized for NGS data. When applied to short and fragmentary sequences, these tools generate marginal scores and thus cannot distinguish reads sequenced from ncRNAs or other regions. To our best knowledge, trCYK [16] is the only tool that conducts homology search for fragmentary reads sequenced from various types of ncRNAs. However, using it alone tends to incur high false positive rate according to our experimental results.

It is worth noting that although NGS platforms are producing longer reads, many reads sequenced from ncRNAs are still fragmentary. First, many ncRNAs are very long, including mRNA-like long ncRNAs [61], 16s rRNAs [55], etc. Second, the biogenesis shows that some types of small ncRNAs are cleavaged into short products (such as mature miRNAs from their precursors). The sizes of these short products are not increasing with read length.

In order to apply existing ncRNA identification tools to NGS data, read mapping or *de novo* sequence assembly tools are usually applied first to connect short reads into contigs. When the reference genome is available, short reads can be mapped back to the reference genome. Existing ncRNA identification tools can then be applied to the blocks containing overlapping reads along the reference genome. When there is no quality reference genome available, which is often the case for metagenomic data and RNA-seq data of non-model organisms, *de novo* sequence assembly tools can be employed first to connect fragmentary reads into contigs. However, using sequence assembly tools as the first step is not always ideal for ncRNA classification.

First, the quality of read assembly deteriorates significantly in complicated NGS data sets [28]. Different sequence assembly tools generate different sets of contigs. There is no consensus on the best assembly tool. Error-containing contigs often affect down-stream analysis.

Second, successful *de novo* assembly requires relatively high sequencing depth, which is difficult to achieve for many ncRNAs. It is shown [62, 63] that the transcription levels of many types of ncRNA genes are low and condition-dependent. Often it is difficult to foresee which ncRNA genes are lowly transcribed. Thus there lacks information to optimize the parameters of *de novo* assembly tools to produce complete or partial ncRNAs of highly divergent expression levels or abundance.

Third, some types of ncRNAs are cleaved during the maturation process (mature miRNAs). The observed reads sequenced from these ncRNAs do not share any overlap and cannot be assembled. Thus, there is a need for an alternative and better ncRNA search tool for NGS data lacking reference genomes.

In this work, we introduce a comprehensive ncRNA classification tool for short reads: *RNA-CODE*, which is specifically designed for ncRNA identification in NGS data sets that lack reference genomes. Given a set of short reads, RNA-CODE classifies the reads into different types of ncRNA families. The classification results can be used to quantify the expression levels of different types of ncRNAs in RNA-seq data and ncRNA composition profiles in metagenomic data, respectively. RNA-CODE integrates secondary structure based homology search with family-specific *de novo* assembly. The parameters of *de novo* assembly tools can be adjusted in a family-specific fashion.

The remaining of this manuscript is organized as follows. The Methods section describes the design rationale of RNA-CODE and the three main stages. The Results section benchmarks RNA-CODE with other ncRNA classification frameworks. We present experiments results on real metagenomic data and RNA-seq data. For the small-scale metagenomic data, we compare RNA-CODE with Metaxa [54] on 16s rRNA read classification. Then, we compare RNA-CODE with *de novo* sequence assembly on small RNA-seq data annotation of a

well-annotated organism.

3.2 Methods

We propose a method that combines homology search and family-specific *de novo* assembly to identify reads sequenced from ncRNAs. In particular, the homology search is applied to both the short reads and contigs produced by assembly programs. This method is designed based on two key observations. First, reads sequenced from ncRNAs tend to share higher sequence and structural similarity with the their native families than reads sequenced from other families. Thus, higher alignment scores by ncRNA homology search tools are expected. In particular, homology search is vital for identifying ncRNAs that go through cleavage and degradation. Reads sequenced from miRNAs are hard to assemble because only reads corresponding to mature miRNAs can be largely captured into RNA-seq data. None or a few can be mapped to other regions of the pre-miRNA due to fast degradation. Figure 3.1 shows the mapping results of reads sequenced from pre-miRNAs obtained from Arabidopsis. No contig or very short contigs can be produced based on the typical read mapping pattern. In addition, this read mapping pattern does not change with increase of expression levels, as shown in the three miRNAs in Figure 3.1. For these types of ncRNAs, applying homology search on short reads directly is indispensable.

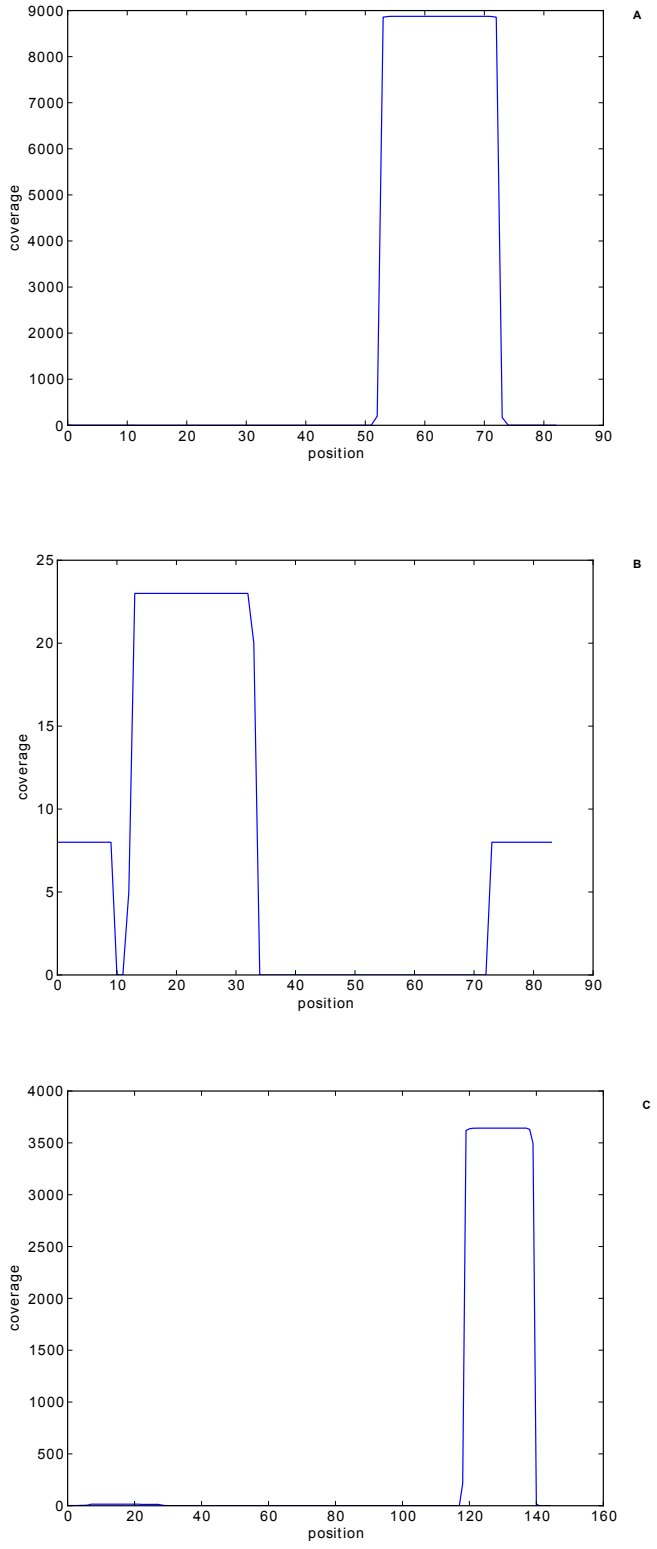

Figure 3.1: Reads sequenced from pre-miRNAs cannot be assembled into contigs. Three different miRNAs show highly different expression levels in the same RNA-seq data. A. mir-156 B. mir-160 C. mir-166

While homology search is important, applying it to short and fragmentary reads may introduce high false positive rate when detecting remote ncRNA homologs (data will be shown in Methods Section). Thus RNA-CODE employs the second observation that true ncRNA reads sequenced from the same gene can be assembled into contigs with significantly high alignment scores against their native families. On the contrary, reads aligned by chance are not likely to be assembled because they tend to share poor overlaps. Both properties are important in boosting sensitivity and accuracy of short reads classification.

RNA-CODE consists of three key stages. First, RNA-CODE coarsely classifies reads into different ncRNA families using both secondary structure and sequence similarity. Then, a family-specific sequence assembly is used to assemble aligned reads into contigs. Because the numbers of reads that are coarsely classified in the first step indicate the expression levels or abundance of ncRNA genes in this family, this step chooses *de novo* assembly parameters (such as kmer size or overlap threshold) accordingly. The produced contigs are generally longer than input reads and thus can be classified into ncRNA families with better sensitivity and accuracy in the last step. For miRNAs which cannot be assembled into contigs, we use their biogenesis-based property and homology search results for classification.

The three-stage workflow with chosen tool for each stage is illustrated in Figure 2.2. Here, we highlight the rationale behind the design of the three stages. The first stage aims to classify a large number of input reads into different ncRNA families with high sensitivity. It employs existing homology search tools. For short and fragmentary reads, this stage can incur high FP rate. Thus, downstream analysis is needed to remove those falsely classified reads. In the second stage, *de novo* sequence assembly tools are employed to assembly classified reads into contigs. The family-specific sequence assembly is expected to produce contigs corresponding to complete or partial ncRNA genes. However, because of the

extremely uneven or low transcriptional levels of many types of ncRNAs or low abundance, a small overlap cutoff or kmer is needed to ensure appropriate connectivity for some families. As a result, some contigs are chimeric or simply consist of randomly aligned reads. The third stage is used to remove the false positives. All contigs are aligned to ncRNA families. Only ones with scores or lengths above given cutoffs are kept. For miRNAs that cannot form contigs, we use stringent homology search scores and known biogenesis-related properties as classification criteria. Every stage will be described in great detail below.

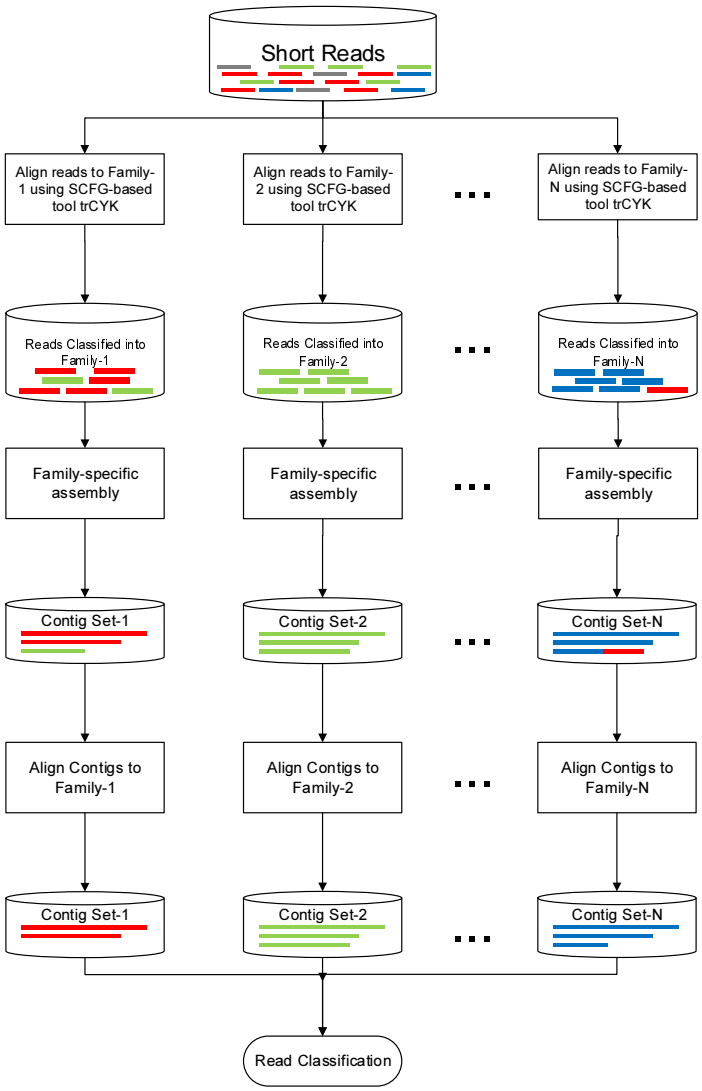

Figure 3.2: The pipeline of RNA-CODE. The pipeline of RNA-CODE. For miRNAs, the output of the first stage (SCFG-based filtration) and the whole pipeline will be used together for reads classification.

3.2.1 Stage 1: SCFG-based filtration

To maximize classification sensitivity, short reads are aligned to an SCFG model built from an RNA family of interest. An SCFG describes not only primary sequence of an RNA family but also its secondary structure formed by base pair interaction. The state-of-the-art implementation of SCFG model is Covariance Model (CM). The software suite Infernal [17] builds a CM on a family of RNA sequences, and searches for homologs using inside-outside algorithms. A CM in Infernal is implemented as a tree-like structure in which each node models a single base or a base pair. Infernal is able to optimally align a sequence to this tree with the highest possible score. Short reads, however, pose challenges to the search algorithms because they are fragmentary sequences in which nucleotides expected to form base pairs could be missing. Due to missing bases, base pairs that could have been aligned to a base-pair node in a parse tree are not alignable any more. As a result, reads sequenced from this family may not be well aligned to the underlying CM. Truncated-CYK (trCYK) [16] is a specialized tool designed for fragmentary sequence search. It performs local RNA alignment against a CM of interest, recovering base pairs that are possibly missing and would otherwise be base paired. For every alignment, a score is provided by trCYK indicating the goodness of alignment. Homologous reads tend to yield higher scores and longer alignments than random reads.

Here we report the performance comparison of two homology search tools that can be applied to short and fragmentary reads. One is the mostly commonly used homology search tool BLAST [15], which relies on sequence similarity only. The second tool is trCYK [16]. The goal is to compare the performance of trCYK with BLAST in classifying ncRNA reads of different lengths. Thus, for read length 25, 30 and 50, we sampled 5000 true reads from

tRNA sequences obtained from Rfam. Another 5000 random reads generated from other RNA families were mixed with true tRNA reads. Seed sequences from Rfam were excluded from the test data. Covariance Model used in trCYK and formatted database used in BLAST were both built from seed sequences of tRNA. We then searched for tRNA reads in the mixed reads using trCYK and BLAST. The performance of both tools is visualized in the ROC curves in Figure 3.3. The figure demonstrates that trCYK has better performance than BLAST. However, both tools have high FP rates, showing the need for further screening.

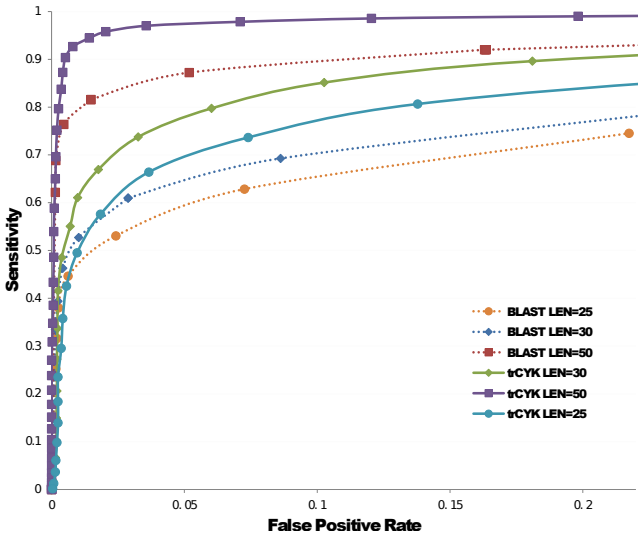

Figure 3.3: ROC curves of short reads classification using trCYK and BLAST. ROC curves of short reads classification using trCYK and BLAST. Sensitivity measures the ratio of correctly found true tRNAs to the total number of true tRNA reads. False positive rate measures the ratio of falsely found tRNA reads to the total number of false tRNA reads.

Like all alignment programs, a score cutoff is needed to distinguish homologous sequences from others. We set two cutoffs s and l, on alignment score and alignment length, respectively. s and l determine the strength of filtration. A low cutoff will lead to an overwhelming number of negative reads which could significantly slow down the next two stages. A high cutoff, however, will exclude remote homologous reads with poor conservation from further analysis. As trCYK does not provide such thresholds, we considered two strategies to determine the

cutoffs. First, the expected alignment score for a homologous sequence of length L can be used as the cutoff. In order to ensure high sensitivity, the actual cutoff can be smaller than the expected score. Second, Monte-Carlo method can be used to evaluate the sensitivity and FP rate of a score cutoff using a large number of sequences that are generated from both ncRNAs and non-ncRNAs. In this work, we used the second strategy. Figure 3.3 shows that some very short homologous reads have poor alignment scores. As the first stage defines the upper-bound of the classification sensitivity, we chose a loose cutoff $s = 1$ to guarantee that most positive reads can pass the filtration stage. We found that this threshold also applies to reads sequenced from other types of ncRNAs. With the increase of read length, this score threshold needs to be improved as well. As the first stage is designed to achieve high sensitivity, the default cutoff is set to 1. To further increase the sensitivity of filtration, we also accept reads with alignment score s greater than -1 and with alignment length $l \geq 30$ bases.

3.2.2 Stage 2: family-specific *de novo* assembly

For reads that are coarsely classified to a family by the first stage, they will be input to *de novo* assembly tools. Compared to conducting *de novo* assembly on all the reads, the input sizes to assembly tools are significantly reduced. Thus, even memory intensive assembly tools can be applied.

Multiple *de novo* assembly tools exist. Depending on the data properties, such as read length and sequencing error rates, sensible choices can be made. In this work, the *de novo* assembly programs are applied to RNA-seq data of non-model organisms or metagenomic data. Thus, specific properties of these two data should be considered when choosing assembly tools. Unlike genome assembly, highly diverse sequencing coverage is expected in

both data sets. In RNA-seq data, heterogeneous expression levels of ncRNAs contribute to highly diverse sequencing coverage. In metagenomic data, different abundance of ncRNA genes lead to different sequencing coverage. Choosing one set of parameters (such as overlap threshold in overlap graph or kmer size in de Bruijn graph) for the whole data set is not likely to produce optimal results for downstream ncRNA analysis. Thus, the first requirement for the chosen assembly program is that users can adjust the parameters according to the output of the filtration stage. Specifically, although the first stage only coarsely classifies reads into gene families, it can be used to estimate the expression levels or abundance of genes in a family. For families with large number of reads classified, RNA-CODE assumes high sequencing coverage and thus uses stringent assembly parameters. On the other hand, for families with fewer number of classified reads, small overlap cutoffs (in an overlap graph-based assembly tools) or small kmers (in de Bruijn graph) should be used to ensure connectivity for lowly transcribed regions or low abundance genes. Second, many assembly programs removed kmers with low coverage as they may contain sequencing errors. In order to assembly ncRNAs of low expression or abundance, we use an assembly tool that can keep reads/kmers with low coverage.

In this work, for very short reads (read length < 50bp), we applied and compared several assembly programs [64, 65] and chose SSAKE [66] because it delivers better assembly performance on our experimental data. SSAKE is a specialized *de novo* assembly tool for unpaired short reads assembly. It is a graph-based greedy assembler that efficiently assembles millions of short reads following near-linear time complexity. During the assembly process, the 3' end of a contig is extended if its suffix overlaps with the prefix of another read. SSAKE generates contigs progressively by searching all prefixes stored in a hash table for the longest possible prefix-suffix match whose overlap is above a threshold. We modified the codes of

SSAKE to make it accept any overlap cutoff.

To follow standard notation for assembly algorithms, we use k to represent either the kmer size in De Bruijn graph or the overlap threshold in an overlap graph for assembly. The length of overlap threshold k is an important parameter in SSAKE. A higher k usually results in fewer but more accurate contigs. A lower k leads to higher contiguity. But incorrect extension may happen because the probability of a random prefix-suffix match is high. In *de novo* assembly, there does not exist optimal overlap threshold. Although the expression or abundance is not known a prior, we estimate it using the number of classified reads in the first stage and choose k accordingly. In addition, it is observed that using a single-k will lead to suboptimal performance of *de novo* assemblers. In the second stage, a range of k's are chosen and used on short reads assembly. For each k, all reads are used in assembly. The contigs from different assemblies are subsequently pooled together for further analysis.

3.2.3 Stage 3: contig selection

Some randomly aligned reads in stage 1 can be removed by stage 2 because they are not part of any contig. However, due to the loose overlap cutoff or small kmers, some reads can still be assembled into contigs and thus produced by stage 2. There are three types of contigs with reference to an ncRNA gene family: 1) positive contigs that are assembled by reads originated from this family, 2) negative contigs that are assembled from false reads that are not part of the underlying gene family, and 3) chimeric contigs that are formed by both true and false reads. Negative and chimeric contigs can be formed due to small overlaps we allowed in the multiple-k assembly. The probability that a contig is extended with a negative read due to a random prefix-suffix match is high for a small overlap cutoff. Sketches of the three types of contigs are presented in Figure 3.2.3.

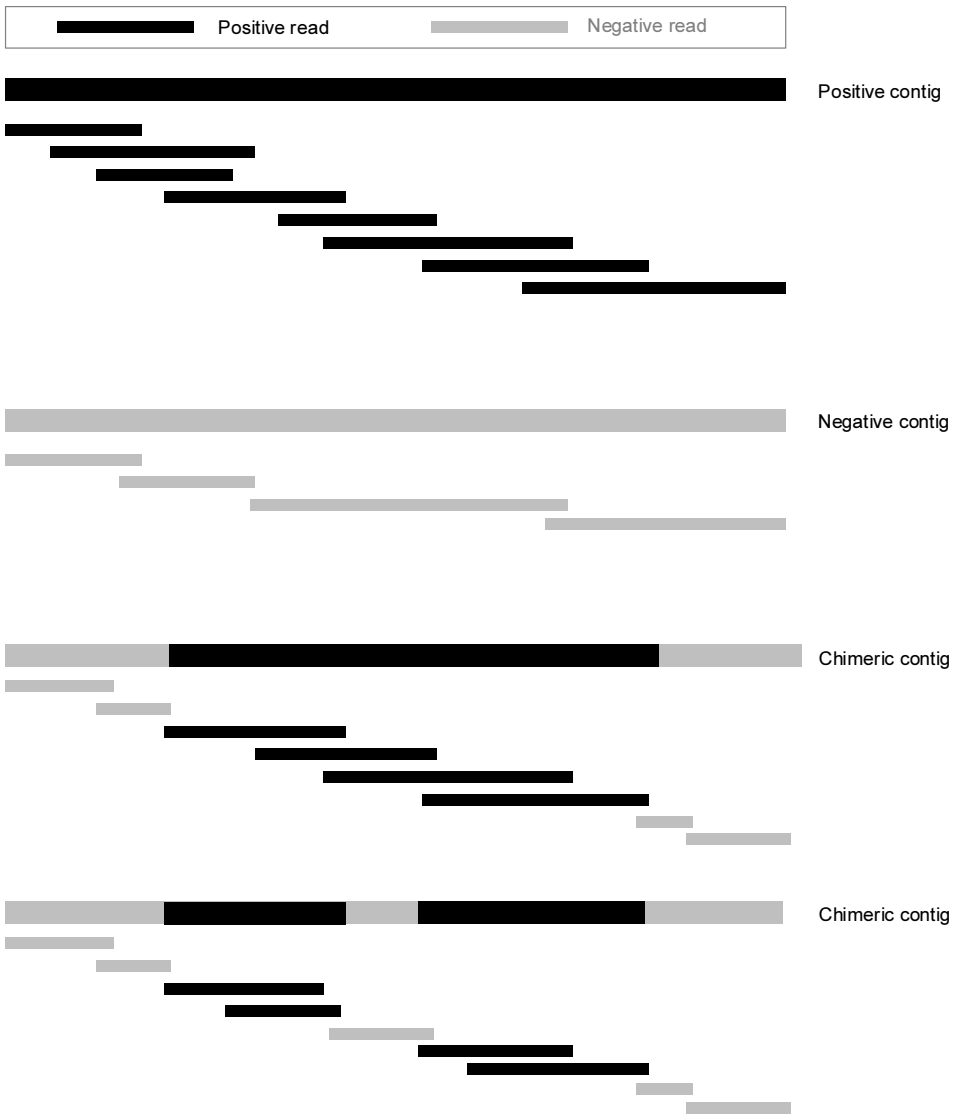

Figure 3.4: Three types of contigs.

To distinguish positive contigs from negative ones, we align contigs to the underlying CM. We chose to use trCYK for the following reasons. 1) Both sequence and structural information of a contig should be utilized. 2) Many contigs may not be complete RNA genes especially when the gene transcription level is low. Thus we need to consider missing bases while aligning contigs to the underlying CM.

After trCYK is applied to all contigs from stage 2, if there exist contigs with alignment scores greater than a pre-determined cutoff, the gene of interest is considered to be tran-

scribed. As trCYK is a local alignment tool, it is common that only part of the contig is aligned to the underlying CM. Thus only reads that assemble the aligned part are classified into this RNA family. This feature could be very effective when multiple correct segments exist in a chimeric contig, although we did not observe such cases in our experiment. Bad segments interleaved by correct ones could potentially be removed.

3.2.4 MIRNA families

Normally, all classified reads need to pass through the entire pipeline. But for miRNA families, as no contigs might be formed, we used two criteria for read classification. First, the alignment score and length of the trCYK alignments in the first stage must pass the predetermined threshold. Second, for all reads that align to the miRNA* region, we examined whether they can form a stem with reads aligned to the mature miRNA region. If not, the reads aligning to miRNA* region will be removed. As a result, many reads that cannot form any contig can be still classified into miRNAs based only on the homology search results in the first stage.

3.3 Results and Discussion

RNA-CODE can be applied to ncRNA classification in both metagenomic data and RNA-seq data of non-model species. To demonstrate the utility of RNA-CODE, we conducted two experiments. In the first experiment, we tested RNA-CODE on identifying reads sequenced from 16s rRNAs in a small-scale metagenomic data set. It is widely known that 16s rRNA is an important genetic marker for taxonomic identification in metagenomes. Identifying 16s rRNA reads can be used as the first step to assemble full-length 16s rRNAs [54, 29, 56].

This experiment aims to detect reads that are sequenced from 16s rRNAs. In the second experiment, we tested RNA-CODE on annotating reads sequenced from different ncRNA genes including house-keeping RNAs, miRNAs etc. in RNA-seq data of the model organism *Arabidopsis thaliana*.

For the first experiment, the performance of RNA-CODE was benchmarked with Metaxa [54], which is designed for classifying short reads into different rRNA families. For the second experiment, the performance of RNA-CODE was benchmarked with standard annotation pipeline for NGS data, which is *de novo* assembly tools plus existing ncRNA annotation tools.

To evaluate the performance of all tools, we compared the true membership and the predicted membership of reads. Two metrics are used in evaluation: read-level sensitivity and positive predictive value (PPV), which indicates accuracy. For an RNA family of interest, let TP be the set of true positive reads originated from this ncRNA family. Let P be the set of reads predicted to be positive. Sensitivity is defined as

$$sensitivity = \frac{P \cap TP}{TP}.$$

PPV is defined as re appropriate

$$PPV = \frac{P \cap TP}{P}.$$

A good ncRNA identification tool should have both high sensitivity and high PPV.

3.3.1 Detecting reads of 16s ribosomal RNAs

RNA-CODE can detect and discriminate among multiple ncRNA types. This experiment tests RNA-CODE in recognition of one type of ncRNA, 16s rRNA in a metagenomic data set. Various tools exist for 16s rRNA search [17, 54, 67, 68], of which, only Metaxa is designed for short reads. Thus, we benchmark RNA-CODE with Metaxa in this experiment.

3.3.1.1 Data

In order to accurately evaluate the performance of RNA-CODE, we need to know the ground-truth membership of reads in metagenomic data. Thus, we constructed a small-scale real metagenomic data set, for which we knew which reads were sequenced from 16s rRNAs. We obtained human gut microbial metagenomics data from European Nucleotide Archive (http://www.ebi.ac.uk/ena/data/view/ERA000116). The data were sequenced from fecal specimens of obese individual human adults using Illumina Genome Analyser [69]. We selected two pair-ended metagenomics data sets of different read lengths. There were 9,633,603 reads of length 44 in the one dataset and 14,822,431 reads of length 75 in the other. Without knowing the genomes of the species in this sample and their 16s rRNA annotations, we cannot obtain the true membership of all the reads. Thus, we need to construct a small metagenomic data set using reads that can be reliably labeled as 16s rRNAs in fully sequenced genomes. To do this, we first chose species that have whole genomes and 16s rRNA annotations using the species catalog in [70]. Then, all reads that can be mapped to these genomes were included in the small metagenomic data set. In total, 11 strains with complete genomes available were selected from 67 strains designed by Turnbaugh [70]. To determine whether a read was originated from 16s ribosomal genes, we mapped each read back to the genomes of the 11 selected bacteria strains. The read mapping positions and the annotation

of 16s rRNA genes are combined to determine whether a read is sequenced from 16s rRNAs. If a mapped read share at least 80% of bases with an underlying 16s genes, we defined this read as positive. If a read has no overlap with any 16s gene, we defined it as negative. The reads having less than 80% of their bases overlapped with a 16s gene are considered to be ambiguous and were discarded. Additionally, the reads that cannot be mapped to any of the 11 genomes were also discarded, because we did not have enough information of their true membership. The small-scale metagenomic data consists of both positive and negative reads. For the data set of read length 44, there are 606 positive reads and 71993 negative reads, respectively. For the data set of read length 75, there are 379 positive reads and 61086 negative reads, respectively.

3.3.1.2 Experimental results

We applied RNA-CODE to the small-scale metagenomic data sets constructed above for 16s rRNA classification. We then compared RNA-CODE with Metaxa, a specialized tool for detecting reads originated from SSU rRNAs such as 12s, 16s and 18s. Same as RNA-CODE, input to Metaxa is also a set of short reads in FASTA format. Output of Metaxa contains short reads presumably originated from SSU rRNAs. Short reads are categorized by the species or organelles that they are originated from. The categories include bacteria, archaea, eukaryota, mitochondria, and chloroplast. Because specific categories of reads are not concerned in this experiment, we considered a read to be a true positive if it can be categorized in any of the 5 categories.

The above experiments were conducted on reads of 44 bases and 75 bases using default parameters. As displayed in Table 3.1, both tools achieved high specificity for both read lengths. The sensitivity of RNA-CODE out-performed Metaxa for both read lengths.

Specifically, RNA-CODE performed well on shorter reads. As Metaxa does read classification using hidden Markov models (HMMs), short reads tend to produce marginal scores and thus are hard to distinguish from non-rRNA reads. RNA-CODE considers both sequence and secondary structure similarity and is more sensitive for short read classification. For the same reason, as Metaxa relies on HMMs, it is not expected to perform well on other types of ncRNAs that lack strong sequence similarity.

	RNA-CODE		Metaxa	
read length	sen	PPV	sen	PPV
44	0.999	1.000	0.786	1.000
75	1.000	1.000	0.986	1.000

Table 3.1: Performance comparison of RNA-CODE vs Metaxa. Both tools were applied using the default parameters.

3.3.2 NCRNA classification in RNA-seq data

To demonstrate the utility of RNA-CODE on detecting reads sequenced from various ncRNA genes, we conducted the second experiment on real RNA-seq data. RNA-CODE classifies reads into different ncRNA families. For house-keeping ncRNAs such as tRNAs and rRNAs, which contain multiple gene members, the number of classified reads show the overall expression levels of this type of ncRNA. For single-member ncRNA families such as many miRNA families in some species, the number of classified reads quantifies the expression level of this gene. We chose to use RNA-seq of the model species *Arabidopsis thaliana*, which has high-quality genome assembly and gene annotation available in TAIR 10 (http://www.arabidopsis.org), enabling us to determine the true membership of reads with

high confidence.

3.3.2.1 Data

An RNA-seq dataset obtained from NCBI SRA (accession number GSM706704) was used in this experiment. This dataset was sampled from transcriptome of inflorescence tissues of *Arabidopsis thaliana*. The sample was sequenced using Illumina platform and contains 2,327,100 short reads. After removing adaptor sequences and quality trimming, the average length of reads is 23.5, which poses a great challenge for both homology search tools and *de novo* assembly tools. Using ncRNA annotation from TAIR 10, there are hundreds of ncRNAs annotated in Arabidopsis. In this work, we present the results on classifying reads into ncRNAs annotated on chromosome 2 of Arabidopsis.

According to TAIR10 annotation and read mapping results, there are 15 transcribed ncRNAs on chromosome 2 in this RNA-seq data. Out of the 15, there is one miRNA mir-825a that does not have corresponding family in Rfam. In addition, miRBase shows that there are only two sequences annotated as mir-825, which are not enough to train a model. Thus, we excluded this miRNA from our test. After removing this family, we had in total 14 transcribed ncRNAs on chromosome 2 in this data set. These families were used to evaluate the sensitivity of ncRNA classification. The number of mapped reads for the 14 ncRNA families can be found in Table 3.2. In order to evaluate both the sensitivity and accuracy of RNA-CODE, we randomly chose 32 non-transcribed but annotated ncRNA families as negative test data. The non-transcribed families have zero mapped read and are used to evaluate the accuracy of RNA-CODE. Ideally, an accurate ncRNA detection tool should not classify any read in this RNA-seq data into these ncRNAs.

ID in Rfam	gene name	num of mapped reads	num mapped reads(unique)
RF00002	5.8S ribosomal RNA	27984	417
RF00005	tRNA	35602	1339
RF00055	Small nucleolar RNA SNORD96	29	12
RF00073	mir-156 microRNA precursor	8875	13
RF00075	Mir-166 microRNA precursor	3657	18
RF00247	Mir-160 microRNA precursor	31	4
RF00268	Small nucleolar RNA snoZ7/snoR77	7	5
RF00300	Small nucleolar RNA Z221/R21b	79	13
RF00452	mir-172 microRNA precursor	101581	15
RF00647	microRNA MIR164	2746	5
RF00689	microRNA MIR390	933	11
RF00690	microRNA MIR408	322	12
RF00893	microRNA MIR854	1367	313
RF01280	Small nucleolar RNA snoR14	9	8

Table 3.2: Number of reads that are mapped to chromosome 2 of Arabidopsis.

For each of the 46 ncRNA families (14 positive + 32 negative), we used the corresponding SCFG-based models in Rfam 10.1 as input to RNA-CODE. Read mapping results and TAIR10 annotation are used to determine the membership of reads. Only if a read has more than 80% of its bases overlapping with an annotated gene, we consider the read to be a member of this gene. Reads with no overlapping bases are unlikely to be valid transcripts of the gene of interest. Such reads are considered to be negative.

3.3.2.2 Experimental results

We evaluate the performance of RNA-CODE from four aspects. First, as the filtration stage of using trCYK is important to the performance of RNA-CODE, we analyze the performance of trCYK in this experiment. Second, we compare the performance of RNA-CODE with *de*

novo assembly tools. Third, we demonstrate the utility of using multiple overlap thresholds in RNA-CODE. Finally, we present case studies for miRNA genes, which produce reads that share no overlaps.

3.3.2.2.1 Performance of filtration In this experiment, we report the performance of the filtration stage for different types of ncRNAs. The first stage of RNA-CODE uses trCYK to coarsely classify reads into different families. Only reads that pass this filtration stage will be further analyzed. Thus, the filtration stage determined the upper bound of the sensitivity of RNA-CODE. A low score cutoff is used to keep high sensitivity. The price paid, however, is low specificity, as displayed in Table3.3. trCYK did not show good discriminative power on some genes because the transcripts sequenced from these genes are not well conserved and cannot form statistically significant alignments with the underlying CM. For example, a majority of transcripts originated from *Small nucleolar RNA Z221/R21b* had alignment scores lower than the defined threshold due to poor conservation. Thus most reads cannot pass the filtration.

gene name	sensitivity	PPV
5.8S ribosomal RNA	0.891	0.415
tRNA	0.882	0.5
Small nucleolar RNA SNORD96	1	0.018
mir-156 microRNA precursor	1	0.039
Mir-166 microRNA precursor	1	0.007
Mir-160 microRNA precursor	1	0.012
Small nucleolar RNA snoZ7/snoR77	1	0.003
Small nucleolar RNA Z221/R21b	0.077	0.001
mir-172 microRNA precursor	1	0.015
microRNA MIR164	1	0.002
microRNA MIR390	1	0.011
microRNA MIR408	1	0.018
microRNA MIR854	0.721	0.447
Small nucleolar RNA snoR14	1	0.007

Table 3.3: Filtration statistics.

3.3.2.2.2 Performance comparison with SSAKE Reads that are classified into each family are used as input to *de novo* assembly programs. For N input families, N *de novo* assembly programs can be run in parallel. A number of *de novo* assembly tools are available. However, due to short read length and low coverage for many types of ncRNAs, some popular tools such as Velvet [64] only produces a few contigs. We empirically compared several *de novo* assembly tools and chose SSAKE 3.8 for two reasons. First, it produced more contigs

than others. Second, the source codes of SSAKE is relatively easy to modify to address specific needs for this data. SSAKE was designed for short reads assembly and the minimum length of an input read is 22 bases. In this data, there exist reads of less than 22 bases. So we customized SSAKE to make it accept reads as short as 19 bases. SSAKE requires a minimum overlap of 16 bases when extending contigs. When gene expression level is low, the overlap between positive reads is often lower than 16 bases. To assemble reads of lowly transcribed genes, we customized SSAKE to extend a contig when its suffix has more than 5 overlapping bases with a prefix of a read. We set *Minimum Number of Reads Needed to Call a Base During an Extension* to be 1, as opposed to 2, the default value. The rationale of using 1 is that in the transcripts of poorly expressed genes, base coverage is low. It is rare to find duplicates of a read when extending with this read.

The chosen assembly tool is run for each family separately. Thus, family-specific assembly parameters can be chosen. In particular, the overlap threshold of SSAKE can be adjusted according to the number of classified reads by trCYK. Although the number of classified reads is not an accurate indication of depth of read coverage, it is preferred to choose a small overlap threshold if the number of classified reads is small. In addition, it has been shown that using multiple kmers can improve RNA-seq assembly [65]. As described in our method, RNA-CODE also uses multiple overlap thresholds. Thus, for SSAKE, we allow overlap from 6 to 16 if the number of classified reads by trCYK is less than 10,000. Otherwise, we use 10 to 20. Users can adjust these parameters according to any known knowledge.

The performance of RNA-CODE on the 14 transcribed families is listed in Table3.4. For the 32 control families, which were not considered to be transcribed, RNA-CODE did not find any reads, yielding 0% false positive rate. This indicates that RNA-CODE can successfully distinguish transcribed families from un-transcribed ones. Table 3.4 also includes

the performance comparison with SSAKE. More specifically, all reads are used as input to SSAKE. Then, all contigs produced by SSAKE are searched against input ncRNA families using trCYK. The comparison shows that RNA-CODE yielded better performance on all genes except for *5.8S ribosomal RNA*. Without the first stage, a large number of negative reads may be assembled together with positive reads and form chimeric contigs. In chimeric contigs, positive reads could be interleaved by negative reads, making the alignment score between itself and the underlying CM low.

gene name	RNA-CODE		SSAKE		single-k	
	sen	PPV	sen	PPV	sen	PPV
5.8S ribosomal RNA	0.778	0.95	0.99	0.884	0.999	0.978
tRNA	0.437	0.984	0.274	0.996	0.044	1
Small nucleolar RNA SNORD96	1	1	0.857	1	0.903	1
mir-156 microRNA precursor	0.929	1	0.786	1	0	N/A
Mir-166 microRNA precursor	1	0.947	0.944	1	0.996	1
Mir-160 microRNA precursor	0.75	0.214	0	N/A	0	N/A
Small nucleolar RNA snoZ7/snoR77	1	1	0	N/A	0	N/A
Small nucleolar RNA Z221/R21b	0	N/A	0	N/A	0	N/A
mir-172 microRNA precursor	1	1	0.733	1	1	1
microRNA MIR164	1	1	1	1	1	1
microRNA MIR390	0.909	1	0.909	1	0.996	1
microRNA MIR408	1	0.75	1	1	1	1
microRNA MIR854	0.599	0.978	0.468	0.698	0.415	1
Small nucleolar RNA snoR14	0.25	1	0	N/A	0	N/A

Table 3.4: Performance of RNA-CODE (multiple-k), SSAKE, and RNA-CODE (single-k) on transcribed ncRNA families.

The reason why RNA-CODE was out-performed on *5.8S ribosomal RNA* is that many short reads originated from *5.8S ribosomal RNA* did not pass the filtration due to poor conservation. As a result, the overall sensitivity is lower than *de novo* assembly tools.

3.3.2.2.3 Using multiple overlap thresholds improves performance of RNA-CODE Multiple-k approach is another important component in the pipeline. To demon-

strate the effectiveness of this approach, we compared RNA-CODE using multiple-k and single-k approach. Sensitivity of RNA-CODE using multiple-k is generally better than using single-k, except for *5.8S ribosomal RNA* and *MIR-390*. We used the default k value (i.e. 16) in single-k approach. Many contigs assembled from short reads originated from lowly expressed genes were not recovered in the single-k approach; since overlap between two positive reads may be less than 16 bases. However, using multiple overlap thresholds may also introduce chimeric contigs, which explains the worse performance of RNA-CODE using multiple k than single k on *5.8S ribosomal RNA* and *MIR-390*.

3.3.2.2.4 Performance of microRNA families For eight transcribed miRNA genes, RNA-CODE performs better in six of them and has the same sensitivity and PPV for the other two. This is expected because miRNA reads usually cannot be assembled. In this data set, because of the extreme short reads, some of them are assembled into the mature miRNA and thus are able to be output by SSAKE. For example, reads of length between 19 and 22 are assembled into the mature miRNA of length 25 for mir-156. For most RNA-seq data that have longer reads after quality trimming, *de novo* assembly tools will not be able to assemble them into contigs. Thus, using both homology search and *de novo* assembly is important to generate a comprehensive catalog of ncRNAs.

3.4 Conclusion

We presented an ncRNA classification tool that can determine the membership of reads that are sequenced from ncRNA genes. By combining homology-based ncRNA search method and family-specific *de novo* assembly, we can classify reads into different types of ncRNAs, including those that cannot be assembled because of cleavage and degradation. This tool can

be applied to NGS data that do not have quality reference genomes, such as metagenomic data and RNA-seq data of non-model organisms.

RNA-CODE relies on trCYK as the ncRNA homology search tool for very short reads. When reads are longer, more efficient ncRNA homology search tool such as Infernal [17] can replace trCYK. For very short reads, trCYK is still the best choice in order to yield high sensitivity.

Chapter 4

Reconstructing 16S rRNA genes in metagenomic data

4.1 Introduction

Microbes are ubiquitous species existing in all environments on earth, including extreme conditions [71]. From the desert to acid waste water, from pine forest soil to mine drainage, they sustain themselves using various mechanisms [72]. Human bodies are also habitants of microbes. It is estimated that there are 10^4 bacterial cell inhabiting on our body, which is 10 times more than our own cells [73] [74]. Human life as well as the entire ecosystem are profoundly affected by microbes. There is a strong need to understand the function of microbial community and how they interact with the environments where they inhabit. It has been discovered that function of microbial community is defined by its composition and diversity [75]. In particular, metagenomic data, which contains sequenced DNA reads of a large number of uncultured microbial species from a wide range of environmental samples, provide a unique opportunity to thoroughly analyze microbial species that have never been identified before.

A commonly adopted approach for analyzing the microbial species in environmental samples is to conduct comparative analysis of ribosomal RNA sequences [76]. The use of rRNA for microbial phylogenetic analysis had become such a relied-upon methodology that by

2008, 77% of all INSDC [77, 78, 79] bacterial DNA sequence submissions described an rRNA gene sequence [80]! 16S rRNA reads from metagenomic studies provide a source of sequences that is not subject to PCR primer bias and therefore covers taxa that might be missed by existing popular primer sets [81]. A fundamental step in using rRNA to analyze the microbial species is to recover the rRNA genes from the large amount of reads in metagenomic data sets, which is the goal of this work. The rRNA genes are a patchwork of hypervariable (rapidly evolving) and universally conserved regions, complicating phylogenetic analysis of rRNA genes in metagenomes both by the lack of usable phylogenetic signal in many unassembled reads and by the difficulty in separately assembling genes for the individual populations in a matagenome. In addition, the massive data volume, short read length, skewed species abundance, and high similarity of 16S rRNAs genes all make rRNA recovery in metagenomic data very difficult. Currently, there is a rapid accumulation of metagenomic data from a wide range of environmental samples. For example, there are roughly 24,367 Gbases of data from Human Microbiome Project (HMP) alone. A tool that can efficiently and accurately recover rRNAs from the large amount of data is in great need for analyzing microbial composition in the underlying samples.

Existing pipelines for annotating rRNAs in metagenomic data can be divided into two groups. The first type of pipelines relies on existing *de novo* assembly tools to output assembled contigs, which are then used as input to available genome-wide rRNA search tools. There are various bulk metagenomic assembly programs [82, 83, 84, 85, 86, 87, 88], which intend to recover individual genomes in an environmental sample. Metagenomic assembly is computationally difficult [83]. A recent review [89] summarized the disadvantages of existing de novo assembly. First, the quality of read assembly deteriorates significantly in complicated NGS data sets. In particular, previous work shows that metagenomic sequencing of high-

complexity microbial communities results in little or no assembly of reads [90, 89]. Second, successful de novo assembly requires high sequencing depth, which is difficult to achieve for all rRNAs in one sample. In addition, our goal is to detect rRNAs in metagenomic data while much of bulk *de novo* assemblies consist of other genomic regions. Thus, the commonly used pipeline of combining bulk metagenomic assembly and genome-wide rRNA detection tools is convenient but not optimized for rRNA detection in metagenomic data. The second type of pipelines avoids bulk metagenomic assembly and incorporates properties of rRNAs [29, 56]. The most promising one is perhaps EMIRGE [29], which uses an expectation maximization approach along with a set of reference gene sequences to assemble rRNA genes from metagenomic data. The method has been used to assemble complete genes from organisms at apparent phylum-level diversity to known cultured organisms [91]. However, EMIRGE requires a large number of known rRNA genes for read mapping and does not separate rRNA genes from closely related organisms.

Therefore, lacking reference genomes, recovering full-length rRNAs from a large number of short and error-prone reads is still an outstanding challenge. In this work, we propose and implement a targeted rRNA assembly program REAGO [1] that is optimized for rRNA gene recovery in metagenomic data. It has three advantages comparing with existing methods. First, it significantly reduces the problem size by first discarding reads that are not likely sequenced from rRNA genes. Secondary structure-aware homology search is adopted in this step to recognize rRNA reads, which will be kept for downstream analysis. Second, paired end information is carefully applied to guide gene assembly and thus distinguish rRNAs from different species. Third, incorporating profile-based homology search, scaffolding step is used to connect contigs with low coverage and thus is able to recover rRNAs in species of

[1] Scoure code is available at https://github.com/chengyuan/reago

low abundance. We applied our tool to both synthetic and real metagenomic data sets and benchmarked its performance with several other programs. The experimental results show that our tool competes favorably in recovery rRNA genes in metagenomic data sets.

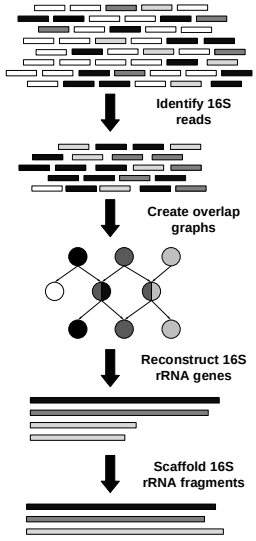

Figure 4.1: Pipeline of the 16S rRNA gene assembly. Short black and gray bars represent reads originated from different 16S rRNA genes. Short white bars represent reads from non-16S regions. Long bars represent contigs assembled from short reads.

4.2 Method

4.2.1 Overview of REAGO

Figure 4.1 is a schematic representation of the pipeline, which contains four stages. First we identify 16S rRNA reads from the original metagenomic dataset using secondary-structure aware homology search. The majority of non-16S reads are discarded in this process, significantly reducing the problem size. Second, REAGO constructs overlap graphs. Various graphs reduction techniques are applied to remove possible sesquencing errors and prepare the graph for efficient assembly. The third stage assembles reads into contigs by our path finding algorithm. The path finding procedure is guided using paired-end information and

aims to avoid generating chimeric 16S rRNA genes by maximizing Weighted Paired-End Match Score (WPEMS) (see Section4.2.5). Finally, paired-end information again is used to scaffold incomplete 16S fragments, if any, into longer contigs or full-length genes.

4.2.2 16S rRNA reads identification

In our method, we first conduct homology search to identify reads originated from 16S rRNA genes. To utilize both the sequence and structural conservation of 16S rRNA genes, we align metagenomic reads to a Stochastic Context-Free Grammar (SCFG) based model [92], which is trained from a selected set of 16S rRNA sequences. The model describes not only primary sequence conservation of a gene family, but also its secondary structure conservation. The-state-of-the-art implementation of SCFG is covariance model (CM) in Infernal [17]. For a gene family, Infernal first builds a CM from a set of training genes from this family and classifies queries by aligning them to the model. A CM is implemented in a tree-like structure in which each node models a single base or a base pair. Inside-outside algorithm [92] is then used to optimally align a query to the CM by maximizing alignment scores. Higher alignment scores are likely to be assigned to reads originated from the family while reads from other genes tend to receive low scores. Fragmentary sequences pose great challenges to the alignment algorithm since structural information in short reads is likely to be partially missing. As a result, such short reads tend to received marginal alignment scores and are not identified. Infernal handles this problem by recovering possibly missing bases while performing the inside-outside algorithms. Thus, short 16S rRNA reads can still receive significant alignment scores.

BLAST [15] is also available for 16S read identification. By aligning reads with the reference 16S rRNA database, rRNA reads may be recognized using BLAST alignment scores

or E-values. We choose to use SCFG-based homology search over BLAST for two reasons. First, BLAST conducts homology search based on sequence similarity and will miss reads lacking primary sequence conservation. Commonly used as the phylogenetic marker gene [93], 16S ribosomal RNAs share high sequence similarity on many regions across different species. However, there also exist a number of variable regions where secondary structures are better conserved than primary sequences. Secondary structural information could then be very helpful to provide additional evidence for 16S sequence identification. A number of studies have demonstrated the advantages of incorporating secondary structural information in various types of non-coding RNA homology search [19]. Experiments were conducted to compare BLAST and CM-based approaches [16] on identifying short rRNA reads. The results indicate that BLAST tends to miss short rRNA reads that are sequenced from non-conserved regions. CM-based approaches generally improve the identification accuracy by including secondary structure information. The second reason behind choosing SCFG-based homology is that the single SCFG-based model provides a convenient reference for deciding the relative positions among contigs during the scaffolding stage. We use the alignment positions of reads inside a contig to determine the contig's rough position along an SCFG model.

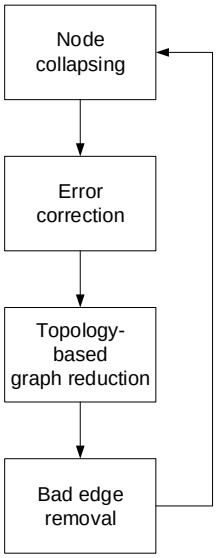

Figure 4.2: Graph reduction is conducted iteratively until there is no change on the graph.

4.2.3 Overlap Graph Creation and Graph Pruning

All reads that are possibly sequenced from rRNA genes are used to construct an overlap graph for assembly. An overlap between two reads is formed if the suffix of a read matches the prefix of another read. A straightforward overlap detection method performs pairwise comparison among all reads, requiring $O(n^2)$ comparisons. Efficient implementations of overlap graphs based on data structures such as hash table and BTW [94, 95] exist to handle NGS data. We choose Readjoiner [94], which provides a set of time and space efficient algorithms for detecting all prefix-suffix matches among a set of reads. In created overlap graphs, each vertex represents a read and each edge represents a prefix-suffix match of size at least l, a pre-determined overlap threshold. l has high impact on complexity and connectivity of graphs. Small l tends to increase the connectivity, but also complexity, of the overlap graph. Larger l is likely to produce less tangled graph, but can possibly miss connections between reads from lowly sequenced regions. Note that transitive edges are automatically removed

in the output by Readjoiner.

The original graphs generated from the output of Readjoiner could be very complex because of the large data size, sequencing errors, and highly similar regions shared by different genes. We apply an iterative graph pruning procedure, as depicted in Figure 4.2, to gradually simplify the graph at each iteration. The procedure terminates when the graph stop changing. Below we detail each stage.

4.2.3.0.5 Node collapsing The original overlap graph tends to have chains of linearly connected vertices. In such chains, each vertex has only a single in-coming edge and out-going edge. Such vertices can be merged without loss of reachability.

4.2.3.0.6 Alignment-based error correction Sequencing errors and highly similar regions shared by different genes can contribute to a large number bifurcations, greatly complicating the graph. Error correction in metagenomic data is an unsolved problem. Rare reads may come from lowly abundant genes rather than contain sequencing errors. Nevertheless, we still follow the error correction rationale commonly used in *de novo* genome assembly and correct bases in rare reads. As shown in Figure 4.3, we applied a heuristic but efficient alignment-based error correction to two types of bifurcations. Reads or contigs from sibling nodes $V_1, V_2, \cdots, V_n$, which share the same predecessor or successor, are aligned. Specifically, based on the known overlaps with the contig in the common predecessor or successor, the contigs in the sibling nodes will be aligned first. Then the reads inside the contigs can be aligned using their positions inside the contig.

For each column in the read alignment, a base is corrected if it is overwhelmingly outvoted by other bases that are aligned to it. An assumption made here is, if a base is sequenced multiple times, it is correctly sequenced majority of the time. A base a is corrected into a',

if and only if the number of a' is at least τ times more than the number of a. It is worth noting that this strategy could potentially incorrectly mutate bases from lowly abundant genes, since the number of those bases could be out-voted by bases from the related and abundant genes. In order to make our tool more practically useful, we sacrifice some accuracy for assembly efficiency. The value of τ is customizable so the user can change the strictness of error correction according to different datasets by adjusting the value of τ. A larger τ tends to yield more conserved error correction. A smaller τ tends to correct all erroneous bases, but may also falsely modify correct bases into incorrect ones, especially in regions with low coverage.

An example is depicted in Figure 4.4. Sequences represented in Vertices V_2 and V_3 are very similar and only differ in one base. The bifurcation may be caused by sequencing error or simply represent similar regions from highly related species. Reads in both vertices are aligned. In the highlighted column, the number of base T in V_3 is far less than that of the base A in V_2, so we mutate the T into A then merge V_2 and V_3 in to a single vertex.

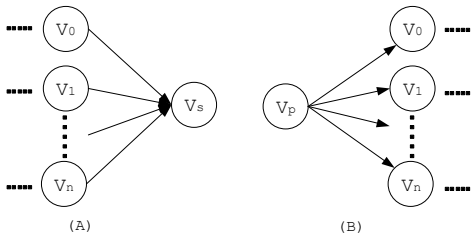

Figure 4.3: Two types of bifurcation where error correction is applied.

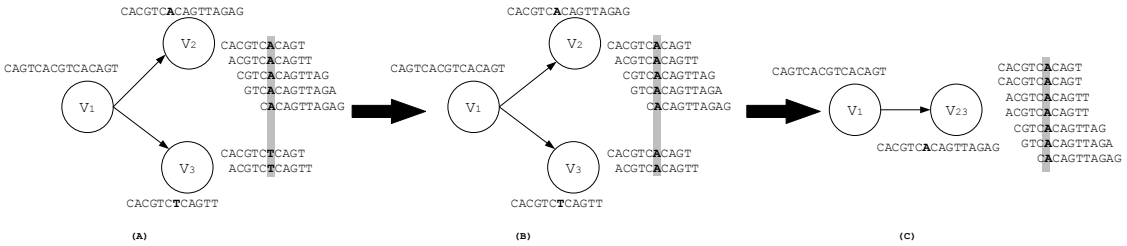

Figure 4.4: An example of error correction (applied on V_2 and V_3). The sequence represented by each node is given beside the node. (A) Ungapped alignment of reads from bifurcating vertices. (B) Mutate rare bases. (C) Remove bifurcation.

4.2.3.0.7 Topology-based graph reduction Alignment-based error correction is only applied to nodes sharing the same predecessor or successor. Following existing assembly methods, we continue to conduct topology-based graph reduction, as depicted in Figure 4.5. We examine all tips and bubbles and remove them if certain criteria are satisfied. Tips are vertices with one end disconnected and must have at least one sibling. Bubbles are formed by two vertices sharing the same predecessor and successor. Tips and bubbles are formed primarily due to sequencing errors and highly similar region shared among closely related species. A tip can be simply dropped from the graphs if two criteria are satisfied. First, the sequence represented by the tip are shorter than a user-defined value. Second, the number of reads collapsed in the tip should be less than a user-defined value. A bubble is resolved by removing the vertex containing less reads if the contigs represented by both vertices share a sequence similarity greater than a threshold, which is set to 98% by default. Local gapped alignment is generated between contigs represented by vertices forming the bubbles. Similar to the bifurcation removal procedure, the tip and bubble removal could potentially remove contigs from low abundant genes. So we also allow the threshold to be adjusted.

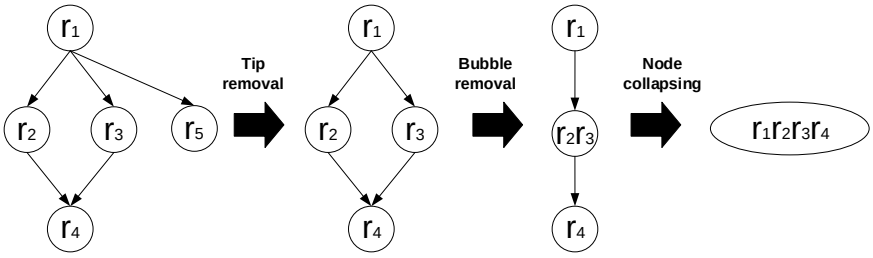

Figure 4.5: Topology-based graph reduction.

4.2.4 Bad edge removal

Due to the nature of metagenomic dataset, there may exist multiple species sharing very high sequence similarity. Thus reads originated from different species have high chance to form edges in the graph. In another word, having prefix-suffix match does not necessarily guarantee correct connection. Wrong edges not only increase the complexity of graph but also lead to chimera.

Thus after graph reduction, we applied a Nave Bayes Classifier, similarly to the RDP classifier [96], on each vertex and approximately annotate it with one or more genera. If the vertices on either of an edge do not share any common annotated genus, we remove this edge from the graph. Annotation of each vertex can be obtained by calculating a posterior probability $P(G_i|C)$, where C is the contig the vertex represents and G_i is the potential genus that C is originated from. The probability indicates the likelihood that the C is originated from G_i. If the probability is higher than a threshold, we annotate C with G_i. To calculate $P(G_i|C)$, we first decompose the contig C into a set of k-mers $k_1, k_2, ..., k_n$. Same as the RDP classifier, the default value of k is 8. Then the likelihood $P(C|G_i)$ can be extended as $P(k_1, k_2, ..., k_n|G_i)$. Assuming the independence of k_i's, we can further simplify the likelihood to $P(k_1|G_i)P(k_2|G_i)\cdots P(k_n|G_i)$, in which each term can be pre-calculated

based on the RDP database [96]. The prior probability $P(G_i)$ can also be calculated as the proportion of sequences in G_i to the total number of sequences in the RDP database. The posterior probability $P(C|\bar{G}_i)$, which indicates the likelihood that C is not originated from G_i, can also be calculated in the same way.

The detailed calculation of posterior probability is list as follows.

$$\begin{aligned}
P(G_i|C) &= P(G_i|k_1, k_2, ..., k_n) \\
&= \frac{P(C|G_i)P(G_i)}{P(C|G_i)P(G_i) + P(C|\bar{G}_i)P(\bar{G}_i)} \\
&= \frac{P(k_1, k_2, ..., k_n|G_i)P(G_i)}{P(k_1, k_2, ..., k_n|G_i)P(G_i) + P(k_1, k_2, ..., k_n|\bar{G}_i)P(\bar{G}_i)} \\
&= \frac{P(k_1|G_i)P(k_2|G_i)\cdots P(k_n|G_i)P(G_i)}{P(k_1|G_i)P(k_2|G_i)\cdots P(k_n|G_i)P(G_i) + P(k_1|\bar{G}_i)P(k_2|\bar{G}_i)\cdots P(k_n|\bar{G}_i)P(\bar{G}_i)}
\end{aligned}$$

where

$$G_i = \frac{\text{number of sequences in genus } G_i}{\text{total number of sequences in RDP database}}$$

$$\bar{G}_i = \frac{\text{number of sequences in genus other than } G_i}{\text{total number of sequences in RDP database}}$$

and

$$P(k_j|G_i) = \frac{\text{number of sequences in genus } G_i \text{ containing } k_j}{\text{total number of sequence in genus } G_i}$$

$$P(k_j|G_i) = \frac{\text{number of sequences in genus other than } G_i \text{ containing } k_j}{\text{total number of sequence in genus } G_i}$$

For a vertex, a genus G_i is included into its annotation if $P(G_i|C)$ is greater than a threshold. As most vertices in the graphs represent only short and partial 16S genes, the classifier may not have enough evidence to uniquely and accurately annotate them. As a result, each vertex is generally associated with multiple genera. Yet, base on our observation, the annotation always include the true positive genus. The edge removal algorithm works

better on longer sequences. So we apply another round of graph collapsing, tip and bubble removal, which potentially could generate vertices representing longer sequences.

4.2.5 Guided path finding using paired-end information

We then recover 16S rRNA sequences by finding paths that represent a full or partial rRNA gene. Path finding starts at a vertex with no in-coming edges and terminates at a vertex with no out-going edges. Paired-end information of reads are widely used in many assembly tools for guiding the creation of contigs or scaffolds. The rationale is that two ends of a read pair should to be assembled in the same contig or scaffold. To utilize the paired-end information, a common approach adopted by SOAPdenovo [97], ABySS [98] and ALLPATHS [99] creates graphs from a set of contigs where each vertex represents a contig and an edge is formed between two vertices if more than a certain number of read pairs exist between their reads. Then the graphs are searched, using various constraints and heuristics, to extend contigs into longer scaffolds. Velvet [64], on the other hand, assumes a small variance of insert size distribution and aims to create "long nodes" that are longer than all inserts. The objective function intends to maximize the number of "long nodes" while minimizing the number of read pairs spanning over such nodes. As an extension of Velvet, MetaVelvet [100] uses paired-end information to guide the creation of contigs by checking their consistency. Number of paired-end reads connecting the origin node and the extension node is used to resolve chimeric node candidates.

In our algorithm, we use paired-end information to guide the path finding. At each vertex with multiple successors, we decide which one to visit based on a new metric, weighted paired-end match score (WPEMS), which gives higher weights to paired end reads located in distant nodes than those located in nearby nodes. Suppose (r, r') is a mate pair, which are located

in vertices V and V' ($V \neq V'$), respectively. The WPEMS of this read pair is thus $2^{d(r,r')}$, where $d(r, r')$ is the number of vertices between V and V'. If $V = V'$, we define its WPEMS as 0. The WPEMS of a path P is the sum of WPEMS of all paired end reads in P.

$$WPEMS(P) = \sum_{(r,r') \ in \ different \ vertices} 2^{d(r,r')}$$

When deciding among a set of vertices to visit next, we select the vertex that maximizes the WPEMS of the current path in a greedy way. The rationale behind the design of WPEMS is that only read pairs existing in non-adjacent vertices provide extra evidence for path finding. Read pairs included in the same vertex or spanning adjacent vertices do not provide additional path finding information beyond the existing overlap graph. In practice, the graph pruning procedure leads to many vertices representing relatively long contigs. Thus, for average fragment sizes such as hundreds of bases, the paired-end reads usually exist within the same vertex or vertices connected by an edge already. Figure 4.6 describes a typical example of read pair distribution in a graph. There are usually many paired-end reads spanning adjacent vertices such as from n_6 to n_8. Much less paired-end reads span non-adjacent vertices, such as those from n_1 to n_7 and from n_4 to n_7. Paired end reads located in two adjacent nodes will have WPEMS of 1, because there is no intermediate nodes between the adjacent nodes.

We use the example in Figure 4.6 to explain our path finding procedure. Nodes in the graph are formed by reads from two different 16S rRNA genes A and B, colored in blue and yellow, respectively. Nodes n_3 and n_6 are represented using shaded color since they are formed by reads from common regions of A and B. Genes A and B are represented by $n_1 \rightarrow n_3 \rightarrow n_4 \rightarrow n_6 \rightarrow n_7$ and $n_2 \rightarrow n_3 \rightarrow n_5 \rightarrow n_6 \rightarrow n_8$. Suppose the path finding

process has successfully identified the path $n_1 \rightarrow n_3 \rightarrow n_4 \rightarrow n_6$, which is highlighted in yellow, and needs to choose between n_7 and n_8 as the next node to include. Although n_8 and n_6 share a large number of paired-end reads, WPEMS will choose n_7, the correct node, because of the paired-end reads shared between non-adjacent nodes.

In order to produce all 16S ribosomal RNA sequences, we apply the path finding algorithm on each vertex with no in-coming edges. If the length of a contig represented by a path is greater than a user-defined threshold, we consider the contig as a full-length gene. Otherwise, we include it in the input to the next scaffolding stage. As this approach is a greedy algorithm, the time complexity is very low. The entire path finding procedure take *O(mn)* time to complete, where m is the average out-degree of each vertex and n is the average distance between nodes with no in-coming edges and nodes with no out-going edges.

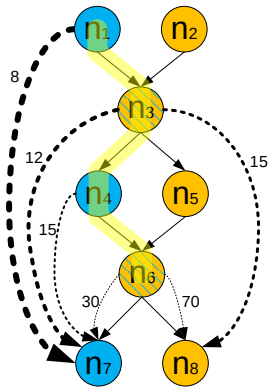

Figure 4.6: Path finding using paired-end information. Solid lines represent overlaps between nodes and dashed lines represent the existence of paired-end reads. The numbers beside dashed lines are the numbers of paired end reads between the corresponding nodes.

4.2.6 Scaffolding 16S rRNA segments

In the path finding stage, contigs longer than a user-defined parameter L, are output directly. Contigs shorter than L are selected for further processing. Short contigs are usually

created due to regionally low coverage of 16S genes in metagenomic data. Reads from lowly sequenced regions failed to create connection to other region due to small overlap. As a result, 16S segments originated from the same gene may be broken apart. To produce full-length 16S sequences, we can utilize the WPEMS, with a minor modification, to scaffold shorter segments. Orientation of segments can be inferred from the reads they contain. As we have the alignment position of each read on the CM used the filtration stage, we can then infer the alignment position of segments on CM. This information allows us to determine orientation of segments as well as their relative position on CM. For two segments S_A and S_B with S_A being in the upstream of S_B (see Figure 4.7), the WPEMS between S_A and S_B is defined as

$$WPEMS(S_A, S_B) = \sum_{(r,r')} 2^{d(r,r')}$$

where (r, r') is a mate pair. r is in S_A and r' is in S_B,

and $d(r, r')$ is total number of vertices after r and before r', in S_A and S_B, respectively

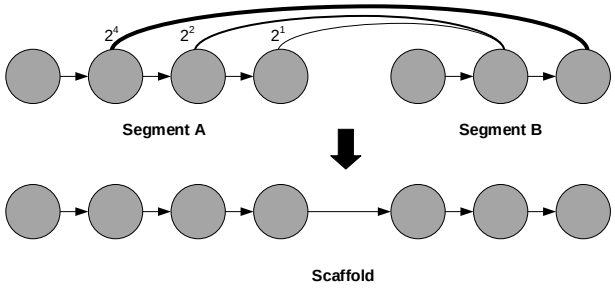

Figure 4.7: Calculate the score between two segments. Arcs represent paired-end match between vertices and thickness of arcs indicate weight of the paired-end match. Actual weights also labeled beside each arc.

We only calculate $WPEMS(S_A, S_B)$ if S_A is on the upstream of S_B and their CM

alignment position overlap. If S_A and S_B are from the same gene, $WPEMS(S_A, S_B)$ tends to be the highest among all $WPEMS(S_A, S_i)$ and $WPEMS(S_j, S_B)$. So we calculate the pair-wise WPEMS in all segments. We then connect S_A and S_B, if

$$WPEMS(S_A, S_B) > WPEMS(S_A, S_i) \text{ for all } S_i \text{ on the downstream of } S_A, \text{ and}$$

$$WPEMS(S_A, S_B) > WPEMS(S_j, S_B) \text{ for all } S_j \text{ on the upstream of } S_B$$

All connected segments with total length longer than L will be output.

4.3 Experimental results

To evaluate the performance of our algorithm, we applied REAGO to three sets of metagenomic data including a simulated data set, a mock community data set, and a human stool metagenomic data set. As the species and their genomes are largely known in the first two data sets, we are able to evaluate the accuracy of rRNA assembly. The third experiment enables us to test the utility of REAGO on a real metagenomic data set. We benchmarked our tool with EMIRGE, a 16S rRNA identification tool, and two bulk metagenomic assembly tools, IDBA-ud [101] and Meta-Velvet [100]. We also compared the performance of the three tools on two sets of inputs, the entire datasets and only true positive reads. For all the tools, we evaluated their performance to reconstruct 16S rRNA sequences at sequence-level and genus-level. We also recorded and compared their running time on the same high performance computing node that has 64 bits CPU with Linux operating system.

Species	abundance
Bacteroides thetaiotaomicron VPI-5482 (BTV)	24.17%
Bacteroides vulgatus (BVG)	4.13%
Chlorobium phaeobacteroides DSM 266 (CPB)	10.02%
Chlorobium phaeovibrioides DSM 265 (CPV)	6.29%
Chlorobium tepidum TLS (CTT)	12.38%
Salinispora tropica CNB-440 (STC)	1.96%
Sulfurihydrogenibium sp YO3AOP1 (SSY)	4.72%
Bordetella bronchiseptica RB50 (BBR)	7.86%
Burkholderia xenovorans LB400 (BXL)	10.02%
Leptothrix cholodnii SP-6 (LCS)	4.72%
Nitrosomonas europaea ATCC 19718 (NEA)	13.75%

Table 4.1: Species abundance.

4.3.1 Experiment 1: simulated metagenomic dataset

To evaluate the performance of REAGO , we first applied it to a simulated metagenomic data set containing reads from 11 species of 8 genera. We used WGSIM [102] to generate 9.6e7 paired-end, 110bp long error-containing reads. The sequencing error rate was set to 2% by default and the insert size was set to 100 with a standard deviation of 10 bases. The 11 selected species have very skewed relative abundance, as shown in Table4.1. The most abundant species is 11 times more than the least abundant species.

To challenge REAGO , we selected some closely related species in the same genus with highly similar 16S rRNA genes. We listed their pair-wise sequence similarity in Table4.2. For example, three selected species in Chlorobium share sequence identity above 93%. Chlorobium phaeobacteroides and Chlorobium phaeovibrioides even share similarity as high as 96%. Reads simulated from these species tend to produce extremely tangled overlap graphs that pose challenges for assembly. Traversing such graphs will result in a large number of paths, and most of which are chimeric assemblies.

	BBR	BTV	BVG	BXL	CPB	CPV	CTT	LCS	NEA	SSY	STC
BBR	-	-	-	-	-	-	-	-	-	-	-
BTV	71	-	-	-	-	-	-	-	-	-	-
BVG	71	**91**	-	-	-	-	-	-	-	-	-
BXL	**91**	72	71	-	-	-	-	-	-	-	-
CPB	76	75	75	75	-	-	-	-	-	-	-
CPV	75	75	75	74	**96**	-	-	-	-	-	-
CTT	75	75	74	74	**93**	**94**	-	-	-	-	-
LCS	**90**	73	72	**90**	76	77	76	-	-	-	-
NEA	88	73	72	89	75	75	74	86	-	-	-
SSY	73	72	72	73	73	73	73	74	74	-	-
STC	76	72	71	76	77	77	76	77	76	76	-

Table 4.2: Pairwise sequence similarity. Bold numbers indicate sequence similarity above 90%.

4.3.1.0.8 Performance of rRNA reads classification using cmsearch Following the pipeline in Figure4.1, cmsearch was applied to the simulated dataset to identify reads originated from rRNA genes. When building the covariance model (CM) for cmsearch, we only used rRNA genes that are not in the simulated data set. Specifically, we downloaded 2,591 bacterial 16S rRNA genes from the RDP website [96]. Then we removed the rRNA genes in the 11 species used for simulation and also the ones in the same genera as those 11 species. As a result, we have 2,219 genes in the training set for building the CM in cmsearch. The performance of cmsearch is quantified using 2 metrics: sensitivity and positive predictive value (PPV). The set of reads sequenced from 16S rRNA genes are true positive (i.e. TP) while the set of reads extracted from other regions are true negative. Let P be the set of reads predicted as positive by cmsearch. Sensitivity is thus defined as $\frac{P \cap TP}{TP}$ and PPV is defined as $\frac{P \cap TP}{P}$. It is worth noting that we only keep reads that can be globally aligned to the covariance model. For reads that are partially sequenced from the rRNA genes and thus produce partial or local alignments, we won't keep them for downstream analysis. Correspondingly, during the performance evaluation, we only use reads that are completely

sequenced from the rRNA genes and non-rRNA regions. The ones sequenced from boundaries of rRNA genes will not be used for computation. For this simulated data set, the sensitivity and PPV of cmsearch in recognizing rRNA reads are both 0.990. Reads originated from 16S rRNA genes were precisely separated from those from other regions of genomes with only a small amount of incorrectly classified reads. The output of cmsearch contains 82,638 reads, which are used as input for overlap graph construction. Compared to the original size of the data set (9.6e7 reads), the problem size is significantly reduced.

4.3.1.0.9 Overlap graph construction The reads classified as rRNA reads by cmsearch were used as input to Readjoiner for efficient overlap graph construction. By default, the overlap threshold was set to 70% of the read length, which is 77 in the simulated data set. Larger overlap may be used when sequencing depth is high for each species, while smaller overlap should be used if some lowly abundant species are present. Smaller overlap, however, tend to yield more tangled graphs. We have tried a range of overlaps from 66 to 99. The results are almost identical on the simulated dataset. The error rate of WGSIM was set to 2%, which means that for every 50 bases generated, 1 error occurs. As a result, the error correction threshold τ was set to 50, indicating that a base will be corrected if there are at least 50 bases of a different kind are aligned to it.

We evaluated the efficiency of our error correction and graph reduction algorithms using the change of graph complexity, which is quantified by the total number of paths in the graph. The increase of the number of different paths implies the increase of the graph complexity. We only record "complete" paths that start at nodes with no in-coming edge and end at nodes with no out-going edges. The original graph contains 16,994,765 paths. After applying our graph reduction procedures, the graph is significantly simplified, containing only 961 paths.

As shown by the assembly results presented in the following section, a majority of the genes have been kept in the reduced graph.

4.3.1.0.10 Assembly performance evaluation Finally we evaluated the performance of our assembly algorithms and compare it with EMIRGE, IDBA-ud, and Meta-Velvet. Metrics used are the number of genes recovered at sequence level, the number of genus recovered, the number of falsely recovered genes, and the running time. For all tools, all contigs longer than 1350 nt are considered to be final output. An assembly is considered to be correct at sequence level if and only if it can be aligned to the true gene with at least 98% of identity.

A genus is correctly recovered as long as one of its genes is recovered. For EMIRGE we did two experiments using two different rRNA gene databases. The first one is its original SSU candidate database excluding just the 11 training genes. The second one uses the original database excluding all genes in the 8 selected genera. In this experiment, there are 83,510 reads fully sequenced from rRNAs. All parameters of IDBA-ud were set as default. For Meta-Velvet, we conducted the multiple experiments with a wide range of kmer length from 20 to 65 and pooled the output together as its final output. As REAGO , IDBA-ud, and Meta-Velvet also output partial genes, we only kept full-length genes.

As displayed in Table4.4, our assembly algorithms demonstrate better performance in reconstructing 16S rRNAs. EMIRGE achieved the same sensitivity as our tool with the first database. But it took much longer running time. For the second database, which excluded all genes in the eight chosen genus as the training set for our tool, it recovered much less genes. Thus, if a genus in a metagenomic dataset does not exist in the SSU candidate database at all, it is possible that the genes from this genus cannot be recovered. IDBA-

tool	# output genes	# genes recovered	# genus recovered	# incorrect assemblies	running time
REAGO	12	11/11	8/8	1	4:53:19
EMIRGE, 16S DB excluding 11 genes	16	10/11	7/8	6	96:12:1
EMIRGE, 16S DB excluding 8 genera	24	5/11	4/8	19	96:14:29
IDBA-ud	416	3/11	3/8	413	16:36:28
Meta-Velvet	1258	1/11	1/8	1257	5:35:30

Table 4.3: Performance of 16S rRNA gene recovery.

tool	# output genes	# genes recovered	# genus recovered	# incorrect assemblies	running time
EMIRGE, 16S DB excluding 11 genes	17	11/11	8/8	5	0:13:01
EMIRGE, 16S DB excluding 8 genera	21	7/11	6/8	14	0:14:10
IDBA-ud	3	3/11	3/8	3	0:5:26
Meta-Velvet	1	1/11	1/8	1	0:0:35

Table 4.4: Performance of 16S rRNA gene recovery on true positive simulated data.

ud was correct only on three genes and Meta-Velvet only identified one. The results show that bulk or general de novo assembly tools are not optimized for recovering 16S rRNAs and thus are not recommended for this task. We also conducted the same experiments on EMIRGE, IDBA-ud and Meta-Velvet using only true positive reads. As displayed in Table 4.4, performance of the three tool increased.

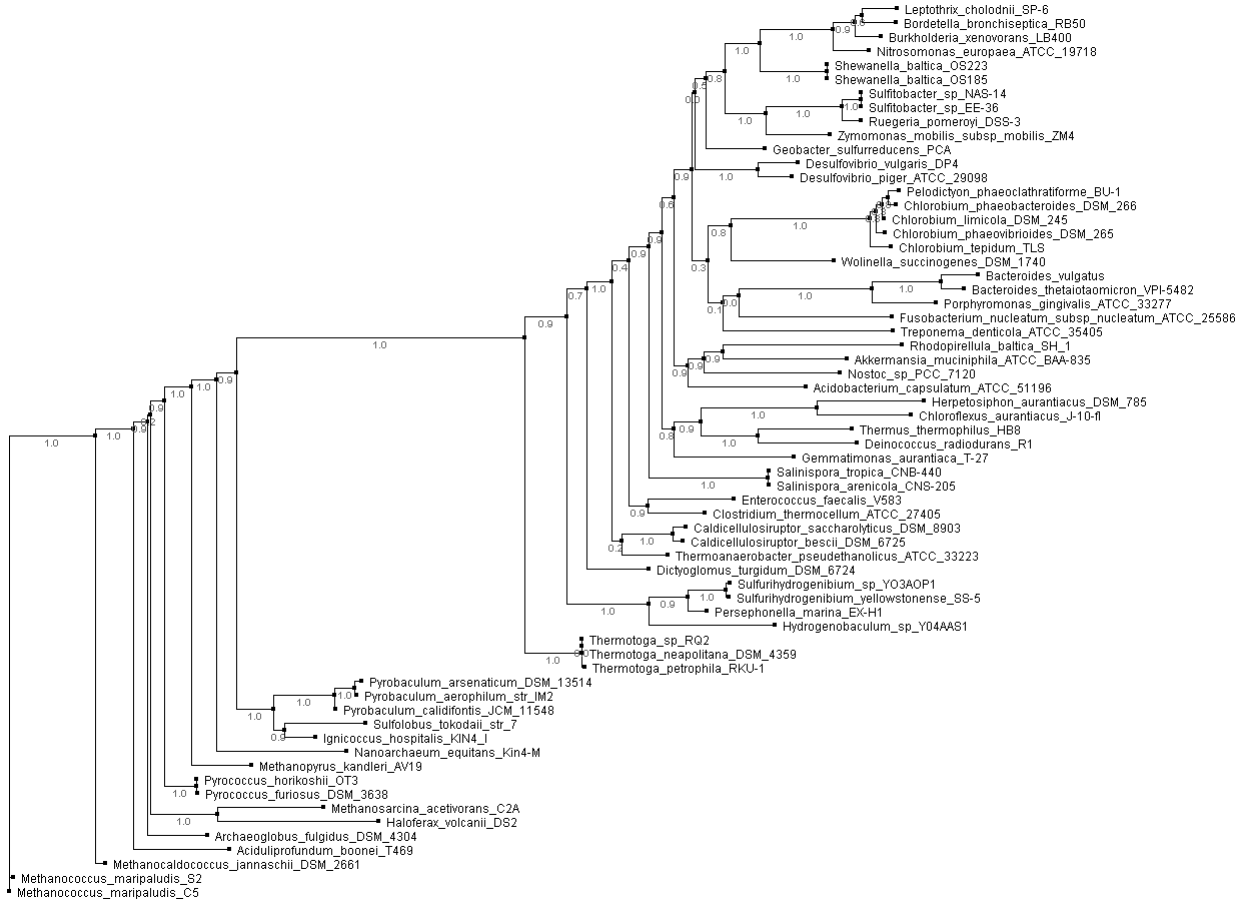

Figure 4.8: Phylogenetic tree of the 64 species in the synthetic metagenomic data.

4.3.2 Experiment 2: synthetic metagenomic data

To further assess the performance of our assembly algorithms, we applied REAGO to a metagenomic dataset (SRR606249) sequenced from mixture of archaeal and bacterial synthetic communities [103], which contain 16 archaeal species and 48 bacterial species, covering 50 different genera. The metagenomic dataset was sequenced using Illumina HiSeq-2000 and contains about 11.1 Gbp in total. Reads in the dataset are all 101-base long and were sequenced in pairs. The annotations of 16S rRNA genes in these species were downloaded from NCBI. High sequence similarity exist among genes in the same genus and genes across different genera, as shown in Figure4.8. For instance, species from Leptothrix, Bordetella,

CM	# output reads	Sensitivity	PPV	running time
remove only exact copies of genes	66,755	0.982	0.962	6:12:13
remove all genes in the 50 genera	66,347	0.976	0.958	6:35:18

Table 4.5: The performance of cmsearch on the synthetic community data.

Burkholdera, and Nitrosomonas share sequence similarity above 97%. Three species under Thermotoga share sequences similarity above 99%. Genes with high sequence similarity lead to very complicated overlap graphs, yielding a large number of paths. Additionally, abundance of species are skewed in this dataset. The most abundant species is over 20 time more than the least abundant species. Skewed abundance level, high sequence similarity, and a large number of error-containing reads pose great challenges for rRNA recovery.

In rRNA read classification, we applied cmsearch to the metagenomic data with default parameters using two CMs. The first CM was trained on all 16S genes from RDP excluding the exact copies of rRNA genes in the 64 species of the synthetic community data. Next, we further removed all genes belonging to any of the 50 genera and trained the second CM. The read mapping results and the annotation of the 16S rRNA genes in the component genomes enable us to determine whether a read is part of a rRNA gene. In this dataset, there are 67,979 reads completely sequenced from rRNA genes. Based on the known origins of the reads, we can evaluate the performance of cmsearch on both CMs using sensitivity, PPV, and running time. As displayed in Table4.5, cmsearch achieved both high sensitivity and PPV with both CMs. Removal of all genes in the 50 genera only slightly decreased the sensitivity and PPV. Running time of cmsearch with eight working threads did not differ much on the two experiments. Table4.5 also shows the significant reduction of problem size by 99.87% (from 54,029,186 reads to less than 68,000 reads). The experiments were conducted using a 2.4GHz CPU and 8GB memory.

tool	# output genes	# genes recovered	# genus recovered	# incorrect assemblies	running time
REAGO	59	58/64	47/50	2	12:28:01
EMIRGE, 16S DB excluding 64 genes	76	42/64	33/50	42	120:30:11
EMIRGE, 16S DB excluding 50 genera	105	19/64	16/50	89	120:25:38
idba-ud	61	39/64	33/50	28	17:26:50
Meta-Velvet	135	4/64	4/50	131	7:56:01

Table 4.6: The performance of rRNA recovery on synthetic community data. Due to near identical outputs and very high sequence similarity among some genes in some genera, the sum of incorrect assemblies and correct assemblies may not sum to the total number of output sequences.

Next, we applied our assembly algorithms on the reads that are classified as rRNA reads by cmsearch. The overlap threshold of REAGO was set to 70 and the error correction threshold was 30. Training sequences of CM contained neither any of the 64 genes nor any gene from the 50 genera. We compared the performance of REAGO with EMIRGE, IDBA-ud, and Meta-Velvet. In this experiment, there are 67,979 reads fully sequenced from rRNAs. Parameters of IDBA-ud were all set as default. Meta-Velvet was run with a wide range of kmer length from 20 to 65. The output were pooled together as its final output. EMIRGE was run twice with two different 16S rRNA databases. The first one is its 16S rRNA database excluding only exact copies of the 64 genes and the second one is its database excluding all genes in the 50 genera. For all tools, we only consider assemblies longer than 1,350nt as the final output. Following the first experiment, we quantified the performance using 4 metrics, including the number of genes recovered at sequence level, the number of genus recovered, the number of incorrect assemblies, and the running time.

We summarized the results in Table4.6. Our tool correctly recovered more genes and genus with far less running time. Even with all genes in the 50 genera removed from the training set of the CM, REAGO can recover 58 out of 64 genes and 47 out of 50 genera. For REAGO , most of the time was spent in running cmsearch and the actual assembly procedure finished within a couple of minutes. The filtration with cmsearch can be greatly accelerated when running in parallel. EMIRGE, on the other hand, recovered less genes and genera

tool	# output genes	# genes recovered	# genus recovered	# incorrect assemblies	running time
EMIRGE, 16S DB excluding 64 genes	82	45/64	36/50	45	0:52:19
EMIRGE, 16S DB excluding 50 genera	103	25/64	20/50	82	0:51:48
idba-ud	60	41/64	34/50	26	0:1:16
Meta-Velvet	65	16/64	16/50	49	0:0:26

Table 4.7: The performance of rRNA recovery on only true positive reads from synthetic community data. Due to near identical outputs and very high sequence similarity among some genes in some genera, the sum of incorrect assemblies and correct assemblies may not sum to the total number of output sequences.

with longer running time. With the removal of all genes in the 50 genera, the performance of EMIRGE significantly deteriorated, indicating its limited ability on recovering 16S genes from unknown genera. It is worth nothing that in this experiment EMIRGE outputs some genes with high sequence similarity but not identical. Thus, several sequences output by EMIRGE may share high sequence similarity with the same true rRNA gene in the data set.

To evaluate the necessity of applying cmsearch to identify 16S reads before the actual 16S gene recovery process, we applied EMIRGE, idba-ud and Meta-Velvet directly on true positive 16S reads. As summarized in Table4.7, the number of recovered genes and genera increased for all tools. Additionally, running time of each tool was significantly reduced. Reads from non-16S did affect the performance of the 16S gene recovery, thus it is very necessary to apply cmsearch before recovering 16S genes for better sensitivity and efficiency.

4.3.3 Experiment 3: human stool metagenomic data

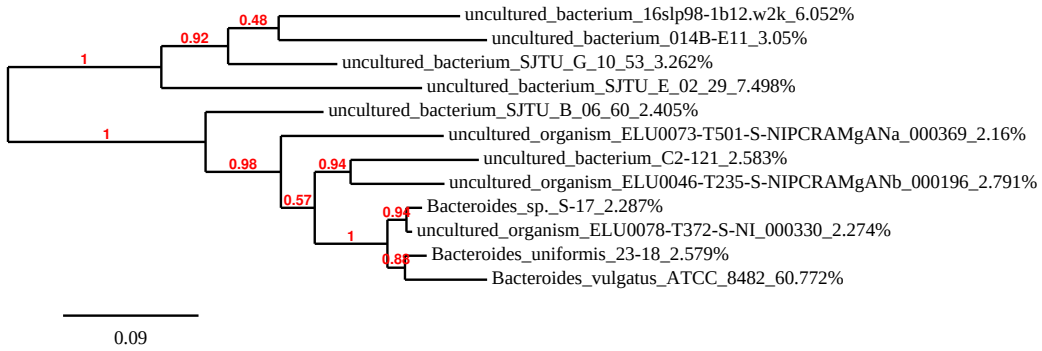

Figure 4.9: Phylogenetic tree of the 12 representative sequences.

CM	number of identified representative sequences	number of other identified sequences	running time
REAGO	7/12	4	6:10:11
EMIRGE	6/12	2	45:12:34
IDBA-ud	1/12	0	6:02:09
Meta-Velvet	0/12	0	2:07:54

Table 4.8: Assembly tool performance on human stool data

In the first two experiments, we evaluated our tool on two data sets with known composition species and annotated rRNA genes. In this section, we describe the application of REAGO to a metagenomic data set sequenced from a human stool sample (SRS015264). The dataset was sequenced using Illumina and contains 31,577,655 100bp paired-end reads. In this experiment, we applied all tools using their default setup without modifying the given training SSU database or the given CM. The overlap threshold for REAGO is still set as 70. Our tool outputs 17 genes and EMIRGE outputs 26 genes. As we don't know the complete composition of the stool metagenomic data set, we compared the output with the known genes identified using read mapping against the reference rRNA databases. Specifically, we mapped the reads to the RDP rRNA database, which contains 1,354,916 16S ribosomal RNA genes. There are 387 genes with at least 200 reads mapped to them. Many of these genes are highly similar. We thus clustered them using CD-HIT-EST [104] using identity cutoff 0.98. 12 clusters are formed by CD-HIT-EST and the phylogenetic tree of their representative sequences with average base coverage is plotted in Figure4.9. There exists a dominant species Bacteroides *vulgatus ATCC 8482*, whose average coverage is 204X, taking 60.77% of the entire dataset. On the other hand, the abundance of other species is very similar with average coverage ranging from 7X to 10X. Among all reads in the dataset, there are 51.1% of them can be mapped to the 12 representative genes. Figure4.9 presents the phylogenetic tree of the 12 representative sequences.

We then compared the output of the tools with the representative sequences of the 12

species	abundance	REAGO	EMIRGE
uncultured_bacterium_16Slp98-1b12.w2k	6.052%	Y	
uncultured_bacterium_014B-E11	3.05%		Y
uncultured_bacterium_SJTU_G_10_53	3.262%		
uncultured_bacterium_SJTU_E_02_29	7.498%	Y	
uncultured_bacterium_SJTU_B_06_60	2.405%	Y	Y
uncultured_organism_ELU0073-T501-S-NIPCRAMgANa_000369	2.16%		Y
uncultured_bacterium_C2-121	2.583%	Y	Y
uncultured_organism_ELU0046-T235-S-NIPCRAMgANb_000196	2.791%		Y
Bacteroides_sp._S-17	2.287%	Y	
uncultured_organism_ELU0078-T372-S-NI_000330	2.274%		
Bacteroides_uniformis_23-18	2.579%	Y	
Bacteroides_vulgatus_ATCC_8482	60.772%	Y	Y

Table 4.9: Species recovered by REAGO and EMIRGE. "Y" indicates the species of the row is identified by the tool labeled by the column title.

clusters and summarized the results in Table4.8. We evaluated their performance using 3 metrics, number of identified representative sequences, number of other sequences, and running time. Our tool successfully identified 7 out of 12 representative sequences and EMIRGE identified 6. Within the 12 representative sequences, the top 3 abundant species are Bacteroides *vulgatus ATCC 8482* and 2 uncultured species SJTU-E-02-29 and 16Slp98-1b12.w2k. Our tool managed to identify all of them while EMIRGE only identified the most abundant species. The detailed outputs of EMIRGE and REAGO are compared in Table4.9. We carefully examined the reasons of REAGO missing 6 representative sequences. During the path finding process, if a path shares its first node with another path, which happens when their sequence similarity is high, only one of the two path will be identified. Our tool failed to identify 6 sequences because of the high sequence similarity between them and the identified sequences as displayed in the phylogenetic tree in Figure4.9.

As described in Table4.8, both tools output a few assemblies that are not from the representative sequences. They may be originated from species that are not in the 12 representative sequence or not even in the RDP database. Applying RDP classifier shows that the uncharacterized assemblies output by EMIRGE and REAGO all belong to Bacteroides and Sutterella.

Chapter 5

Conclusion and future work

Noncoding RNAs play important roles in various important biological functions. Identification of ncRNAs in genomic sequences becomes increasingly important for modern biology. As many families of RNAs are more conserved in their secondary structures than primary sequences, it is necessary to include both pieces of information when creating models. Modeling only primary sequences could lead to loss of sensitivity. With the advent of next-generation sequencing (NGS) technology, massive amount of data has been rapidly accumulating, making many existing algorithms infeasible. As a result, efficient analysis of such data is needed. Based on objectives of algorithms, it is necessary to exclude unrelated data from being analyzed. For graph-based algorithms, the graph size could be very sensitive to the input size. Large amount of input data could potentially lead to high computational cost. Due to the limitation of sequencing technologies, data sequenced is normally fragmentary. Considering sequencing error and uneven sequencing depth, recovery of longer or full-length genes is still a challenging task. With the advance of sequencing technologies, reads with increasing length and lower sequencing error rate could be available. Gene recovery and short reads assembly in general could be easier and more reliable. Paired-end information sometimes could be very useful for gene recovery from NGS data. Especially for many assembly algorithms, it provides additionally information to aid graph traversal.

In the dissertation, a set of algorithms called REAGO is introduced to recover 16S ribosomal RNA from metagenomic data. REAGO first adopts a very effective approach to reduce

the input size. Then it represents all 16S genes in an overlap graph. After a set of graph reduction procedure, it generates a list of 16S rRNA contigs. Short contigs are combined to produce the final output. On both experiments, REAGO out-performed EMIRGE, the state-of-the-art tool, and other two bulk assemblers.

Algorithms in REAGO can be readily extended to identify other types of noncoding RNAs in NGS datasets. They could be especially effective for the families of ncRNAs with variable primary sequence conservation. If the primary sequences of a RNA family are known to be conserved, we could still apply REAGO. However, other bulk metagenomic data assemblers, such as Meta-Velvet and IDBA-ud, should also be good options after we apply the read identification procedure in REAGO. It is worth noting that, Algorithms in REAGO may fail if sequences in a RNA family contain repeats, which may form cycles in overlap graphs, as REAGO is not be able to resolve cycles.

In general, Assembly algorithms for identifying ncRNA in NGS data should be customized to various sequencing platforms, which may produce short reads with various attributes, such as read length, error rate and insert size. Assembly algorithms should be adjusted to these attributes. For example, if the data is obtained from PacBio sequencing platform, which produces very long reads (up to 1500nt) but with high error rate (12.86%). Assembly algorithms should be redesigned focusing on erroneous base correction. Accordingly, the overlap detection using Readjoiner is not feasible anymore. New tools should be developed for this task.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Storz G (2002) An expanding universe of noncoding rnas. Science 296: 1260–1263.

[2] Eddy SR (2001) Non–coding rna genes and the modern rna world. Nature Reviews Genetics 2: 919–929.

[3] Vitreschak AG, Rodionov DA, Mironov AA, Gelfand MS (2004) Riboswitches: the oldest mechanism for the regulation of gene expression? TRENDS in Genetics 20: 44–50.

[4] Durbin R, Eddy SR, Krogh A, Mitchison G (1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. UK: Cambridge University Press.

[5] Eliceiri G (1999) Small nucleolar rnas. Cellular and Molecular Life Sciences CMLS 56: 22–31.

[6] Maxwell E, Fournier M (1995) The small nucleolar rnas. Annual review of biochemistry 64: 897–934.

[7] Fournier MJ, Stuart Maxwell E (1993) The nucleolar snrnas: catching up with the spliceosomal snrnas. Trends in biochemical sciences 18: 131–135.

[8] Ambros V (2004) The functions of animal microrna. Nature 431: 350–355.

[9] Bartel DP (2004) Microrna: genomics, biogenesis, mechanism, and function. cell 116: 281–297.

[10] Lu S, Shi R, Tsao CC, Yi X, Li L, et al. (2004) Rna silencing in plants by the expression of sirna duplexes. Nucleic acids research 32: e171–e171.

[11] Lowe TM, Eddy SR (1997) trnascan-se: a program for improved detection of transfer rna genes in genomic sequence. Nucleic acids research 25: 0955–964.

[12] Schattner P, Brooks AN, Lowe TM (2005) The trnascan-se, snoscan and snogps web servers for the detection of trnas and snornas. Nucleic acids research 33: W686–W689.

[13] An J, Lai J, Lehman ML, Nelson CC (2013) mirdeep*: an integrated application tool for mirna identification from rna sequencing data. Nucleic acids research 41: 727–737.

[14] An J, Lai J, Sajjanhar A, Lehman ML, Nelson CC (2014) mirplant: an integrated tool for identification of plant mirna from rna sequencing data. BMC bioinformatics 15: 275.

[15] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. Journal of molecular biology 215: 403–410.

[16] Kolbe DL, Eddy SR (2009) Local rna structure alignment with incomplete sequence. Bioinformatics 25: 1236–1243.

[17] Nawrocki EP, Kolbe DL, Eddy SR (2009) Infernal 1.0: inference of rna alignments. Bioinformatics 25: 1335–1337.

[18] Kolbe DL, Eddy SR (2009) Local rna structure alignment with incomplete sequence. Bioinformatics 25: 1236-1243.

[19] Nawrocki EP, Kolbe DL, Eddy SR (2009) Infernal 1.0: Inference of RNA alignments. Bioinformatics 25: 1335-1337.

[20] Yuan C, Sun Y (2013) RNA-CODE: a noncoding RNA classification tool for short reads in NGS data lacking reference genomes. PLOS ONE 8: e77596.

[21] Tsiatis AC, Norris-Kirby A, Rich RG, Hafez MJ, Gocke CD, et al. (2010) Comparison of sanger sequencing, pyrosequencing, and melting curve analysis for the detection of¡ i¿ kras¡/i¿ mutations: Diagnostic and clinical implications. The Journal of Molecular Diagnostics 12: 425–432.

[22] Grada A, Weinbrecht K (2013) Next-generation sequencing: methodology and application. Journal of Investigative Dermatology 133: e11.

[23] Liu L, Li Y, Li S, Hu N, He Y, et al. (2012) Comparison of next-generation sequencing systems. BioMed Research International 2012.

[24] Wang Z, Gerstein M, Snyder M (2009) Rna-seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics 10: 57–63.

[25] Handelsman J (2004) Metagenomics: application of genomics to uncultured microorganisms. Microbiology and molecular biology reviews 68: 669–685.

[26] Huang Z, Wu Y, Robertson J, Feng L, Malmberg RL, et al. (2008) Fast and accurate search for non-coding RNA pseudoknot structures in genomes. Bioinformatics 24: 2281-2287.

[27] Huang Z, Malmberg R, Mohebbi M, Cai L (2010) RNAv: Non-coding RNA secondary structure variation search via graph homomorphism. In: CSB Conference Proceedings, CA, USA. pp. 56-69.

[28] Martin JA, Wang Z (2011) Next-generation transcriptome assembly. Nature Reviews Genetics 12: 671–682.

[29] Miller CS, Baker BJ, Thomas BC, Singer SW, Banfield JF, et al. (2011) Emirge: reconstruction of full-length ribosomal genes from microbial community short read sequencing data. Genome Biol 12: R44.

[30] Powers T, Noller HF (1991) A functional pseudoknot in 16s ribosomal rna. The EMBO journal 10: 2203.

[31] Staple DW, Butcher SE (2005) Pseudoknots: RNA Structures with Diverse Functions. PLoS Biology 3: e213.

[32] Gilley D, Blackburn EH (1999) The telomerase RNA pseudoknot is critical for the stable assembly of a catalytically active ribonucleoprotein. PNAS 96: 6621-6625.

[33] Chen JL, Greider CW (2005) Functional analysis of the pseudoknot structure in human telomerase RNA. PNAS 102: 8080-8085.

[34] Wower IK, Zwieb C, Wower J (2004) Contributions of pseudoknots and protein SmpB to the structure and function of tmRNA in trans-translation. the Journal of Biological Chemistry 279: 54202-54209.

[35] Griffiths-Jones S (2007) Annotating Noncoding RNA Genes. Annual Review of Genomics and Human Genetics 8: 279-298.

[36] Klein RJ, Eddy SR (2003) RSEARCH: finding homologs of single structured RNA sequences. BMC Bioinformatics 4: 44.

[37] Lowe T, Eddy SR (1997) TRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. Nucleic Acids Res 25: 955–64.

[38] Cai L, Malmberg RL, Wu Y (2003) Stochastic modeling of RNA pseudoknotted structures: a grammatical approach. Bioinformatics 19: i66-i73.

[39] Rivas E, Eddy SR (2000) The language of RNA: a formal grammar that includes pseudoknots. Bioinformatics 16: 334-340.

[40] Uemura Y, Hasegawa A, Kobayashi S, Yokomori T (1999) Tree adjoining grammars for rna structure prediction. Theoretical Computer Science 210: 277 - 303.

[41] Matsui H, Sato K, Sakakibara Y (2005) Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot rna structures. Bioinformatics 21: 2611-2617.

[42] Kato Y, Seki H, Kasami T (2006) RNA pseudoknotted structure prediction using stochastic multiple context-free grammar. IPSJ Digital Courier 2: 655-664.

[43] Smith JA (2009) RNA Search with Decision Trees and Partial Covariance Models. TCBB 6: 517-527.

[44] Brown M, Wilson C (1996) RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. In: Pacific Symposium on Biocomputing (PSB '96), Hawaii, USA. pp. 109-125.

[45] Zhang S, Haas B, Eskin E, Bafna V (2005) Searching genomes for noncoding RNA using FastR. IEEE/ACM Transactions on Comp Bio and Bioinf (TCBB) 2: 366-79.

[46] Weinberg Z, Ruzzo W (2004) Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. Bioinformatics 20 suppl. 1: i334–40.

[47] Eddy S (2007). HMMER - biosequence analysis using profile hidden Markov models. Http://hmmer.janelia.org/.

[48] Gardner P, Daub J, Tate J, Nawrocki E, Kolbe D, et al. (2008) Rfam: updates to the RNA families database. Nucleic Acids Research 37(Database issue): D136-D140.

[49] Macke T, Ecker D, Gutell R, Gautheret D, Case D, et al. (2001) RNAMotif – A new RNA secondary structure definition and discovery algorithm. Nucleic Acids Research 29: 4724-4735.

[50] Wei F, Stein JC, Liang C, et al (2009) Detailed analysis of a contiguous 22-mb region of the maize genome. PLoS Genet 5: e1000728.

[51] van Batenburg FHD, Gultyaev AP, Pleij CWA, Ng J, Oliehoek J (2000) PseudoBase: a database with RNA pseudoknots. Nucleic Acids Research 28: 201-204.

[52] Jones-Rhoades MWW, Bartel DPP, Bartel B (2006) Micrornas and their regulatory roles in plants. Annual Review of Plant Biology 57: 19-53.

[53] Lu S, Shi R, Tsao CC, Yi X, Li L, et al. (2004) RNA silencing in plants by the expression of siRNA duplexes. Nucl Acids Res 32: e171.

[54] Bengtsson J, Eriksson KM, Hartmann M, Wang Z, Shenoy BD, et al. (2011) Metaxa: a software tool for automated detection and discrimination among ribosomal small subunit (12s/16s/18s) sequences of archaea, bacteria, eukaryotes, mitochondria, and chloroplasts in metagenomes and environmental sequencing datasets. Antonie Van Leeuwenhoek 100: 471–475.

[55] Shah N, Tang H, Doak TG, Ye Y (2011) Comparing bacterial communities inferred from 16s rRna gene sequencing and shotgun metagenomics. In: Pacific Symposium on Biocomputing. World Scientific, volume 16, pp. 165–176.

[56] Fan L, McElroy K, Thomas T (2012) Reconstruction of ribosomal rna genes from metagenomic data. PloS one 7: e39948.

[57] Stricklin Sea (2005) C. elegans noncoding RNA genes, WormBook, ed. The C. elegans Research Community. WormBook.

[58] Ge X, Zhang Y, Jiang J, Zhong Y, Yang X, et al. (2013) Identification of MicroRNAs in Helicoverpa armigera and Spodoptera litura based on deep sequencing and homology analysis. Int J Biol Sci 9: 1-15.

[59] Gruber AR, Neuböck R, Hofacker IL, Washietl S (2007) The RNAz web server: prediction of thermodynamically stable and evolutionarily conserved RNA structures. Nucleic Acids Research 35: W335–W338.

[60] Lowe TM, Eddy SR (1999) A computational screen for methylation guide snoRNAs in yeast. Science 283: 1168–1171.

[61] Mercer TR, Dinger ME, Mattick JS (2009) Long non-coding RNAs: insights into functions. Nature Reviews Genetics 10: 155–159.

[62] Sun G, Stewart CNJ, Xiao P, Zhang B (2012) MicroRNA expression analysis in the cellulosic biofuel crop switchgrass (Panicum virgatum) under abiotic stress. PLoS One 7.

[63] Peng X, Gralinski L, Ferris MT, Frieman MB, Thomas MJ, et al. (2011) Integrative deep sequencing of the mouse lung transcriptome reveals differential expression of diverse classes of small RNAs in response to respiratory virus infection. MBio 2.

[64] Zerbino DR, Birney E (2008) Velvet: algorithms for de novo short read assembly using de bruijn graphs. Genome research 18: 821–829.

[65] Schulz MH, Zerbino DR, Vingron M, Birney E (2012) Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. Bioinformatics 28: 1086–1092.

[66] Warren RL, Sutton GG, Jones SJ, Holt RA (2007) Assembling millions of short DNA sequences using ssake. Bioinformatics 23: 500–501.

[67] Lagesen K, Hallin P, Rødland EA, Stærfeldt HH, Rognes T, et al. (2007) RNAmmer2: consistent and rapid annotation of ribosomal rna genes. Nucleic acids research 35: 3100–3108.

[68] Vilo C, Dong Q (2012) Evaluation of the RDP classifier accuracy using 16s rRNA gene variable regions. Metagenomics 1: 1–5.

[69] Qin J, Li R, Raes J, Arumugam M, Burgdorf KS, et al. (2010) A human gut microbial gene catalogue established by metagenomic sequencing. Nature 464: 59–65.

[70] Turnbaugh PJ, Quince C, Faith JJ, McHardy AC, Yatsunenko T, et al. (2010) Organismal, genetic, and transcriptional variation in the deeply sequenced gut microbiomes of identical twins. Proceedings of the National Academy of Sciences 107: 7503–7508.

[71] Rothschild LJ, Mancinelli RL (2001) Life in extreme environments. Nature 409: 1092–1101.

[72] Konings WN, Albers SV, Koning S, Driessen AJ (2002) The cell membrane plays a crucial role in survival of bacteria and archaea in extreme environments. Antonie Van Leeuwenhoek 81: 61–72.

[73] Savage DC (1977) Microbial ecology of the gastrointestinal tract. Annual Reviews in Microbiology 31: 107–133.

[74] Berg RD (1996) The indigenous gastrointestinal microflora. Trends in microbiology 4: 430–435.

[75] Loreau M, Naeem S, Inchausti P, Bengtsson J, Grime J, et al. (2001) Biodiversity and ecosystem functioning: current knowledge and future challenges. science 294: 804–808.

[76] Woese CR, Kandler O, Wheelis ML (1990) Towards a natural system of organisms: proposal for the domains Archaea, Bacteria, and Eucarya. Proceedings of the National Academy of Sciences of the United States of America 87: 4576-9.

[77] Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW (2010) GenBank. Nucleic Acids Research 38: D46-D51.

[78] Cochrane G, Akhtar R, Bonfield J, Bower L, Demiralp F, et al. (2009) Petabyte-scale innovations at the European Nucleotide Archive. Nucleic Acids Research 37: D19-D25.

[79] Tateno Y, Imanishi T, Miyazaki S, Fukami-Kobayashi K, Saitou N, et al. (2002) DNA Data Bank of Japan (DDBJ) for genome scale research in life science. Nucleic Acids Research 30: 27-30.

[80] Christen R (2008) Global sequencing: a review of current molecular data and new methods available to assess microbial diversity. Microbes and environments JSME 23: 253-268.

[81] Hamady M, Knight R (2009) Microbial community profiling for human microbiome projects: tools, techniques, and challenges. Genome Research 19: 1141-1152.

[82] Namiki T, Hachiya T, Tanaka H, Sakakibara Y (2012) Metavelvet: an extension of velvet assembler to de novo metagenome assembly from short sequence reads. Nucleic Acids Research 40: e155.

[83] Treangen T, Koren S, Sommer D, Liu B, Astrovskaya I, et al. (2013) MetAMOS: a modular and open source metagenomic assembly and analysis pipeline. Genome Biology 14: R2.

[84] Laserson J, Jojic V, Koller D (2011) Genovo: de novo assembly for metagenomes. J Comput Biol 18: 429-443.

[85] Peng Y, Leung HCM, Yiu SM, Chin FYL (2011) Meta-IDBA: a de novo assembler for metagenomic data. Bioinformatics 27: i94-i101.

[86] Luo C, Tsementzi D, Kyrpides N, Konstantinidis K (2012) Individual genome assembly from complex community short-read metagenomic datasets. ISME J 6: 898-901.

[87] Salzberg SL, Sommer DD, Puiu D, Lee VT (2008) Gene-boosted assembly of a novel bacterial genome from very short reads. PLOS Comput Biol 4: e1000186.

[88] Wu Y, Rho M, Doak TG, Ye Y (2012) Stitching gene fragments with a network matching algorithm improves gene assembly for metagenomics. Bioinformatics 28: i363-i369.

[89] Jeffrey AM, Zhong W (2011) Next-generation transcriptome assembly. Nature Reviews Genetics 12: 671-682.

[90] Tringe SG, von Mering C, Kobayashi A, Salamov AA, Chen K, et al. (2005) Comparative metagenomics of microbial communities. Science 308: 554-557.

[91] Wrighton KC, Thomas BC, Sharon I, Miller CS, Castelle CJ, et al. (2012) Fermentation, hydrogen, and sulfur metabolism in multiple uncultivated bacterial phyla. Science 337: 1661-1665.

[92] Durbin R (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge university press.

[93] Woese CR, Fox GE (1977) Phylogenetic structure of the prokaryotic domain: the primary kingdoms. Proceedings of the National Academy of Sciences 74: 5088–5090.

[94] Gonnella G, Kurtz S (2012) Readjoiner: a fast and memory efficient string graph-based sequence assembler. BMC bioinformatics 13: 82.

[95] Simpson JT, Durbin R (2012) Efficient de novo assembly of large genomes using compressed data structures. Genome Research 22: 549-556.

[96] Cole JR, Chai B, Farris RJ, Wang Q, Kulam S, et al. (2005) The ribosomal database project (rdp-ii): sequences and tools for high-throughput rrna analysis. Nucleic acids research 33: D294–D296.

[97] Simpson JT, Durbin R (2012) Efficient de novo assembly of large genomes using compressed data structures. Genome research 22: 549–556.

[98] Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, et al. (2009) Abyss: a parallel assembler for short read sequence data. Genome research 19: 1117–1123.

[99] Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, et al. (2008) Allpaths: de novo assembly of whole-genome shotgun microreads. Genome research 18: 810–820.

[100] Namiki T, Hachiya T, Tanaka H, Sakakibara Y (2012) Metavelvet: an extension of velvet assembler to de novo metagenome assembly from short sequence reads. Nucleic acids research 40: e155–e155.

[101] Peng Y, Leung HC, Yiu SM, Chin FY (2012) Idba-ud: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics 28: 1420–1428.

[102] Li H (2011). wgsim-read simulator for next generation sequencing.

[103] Shakya M, Quince C, Campbell JH, Yang ZK, Schadt CW, et al. (2013) Comparative metagenomic and rrna microbial diversity characterization using archaeal and bacterial synthetic communities. Environmental microbiology 15: 1882–1899.

[104] Li W, Godzik A (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics 22: 1658–1659.