Computational Methods for Identification of Human microRNA Precursors

Jin-Wu Nam, Wha-Jin Lee, and Byoung-Tak Zhang

Graduate Program in Bioinformatics
Center for Bioinformation Technology
Biointelligence Laboratory, School of Computer Science and Engineering
Seoul National University, Seoul 151-742, Korea
{jwnam,wjlee,btzhang}@bi.snu.ac.kr

Abstract. MicroRNA (miRNA), one of non-coding RNAs (ncRNAs), regulates gene expression directly by arresting the messenger RNA (mRNA) translation, which is important for identifying putative miRNAs. In this study, we suggest a searching procedure for human miRNA precursors using genetic programming that automatically learn common structures of miRNAs from a set of known miRNA precursors. Our method consists of three-steps. At first, for each miRNA precursor, we adopted genetic programming techniques to optimize the RNA Common-Structural Grammar (RCSG) of populations until certain fitness is achieved. In this step, the specificity and the sensitivity of a RCSG for the training data set were used as the fitness criteria. Next, for each optimized RCSG, we collected candidates of matching miRNA precursors with the corresponding grammar from genome databases. Finally, we selected miRNA precursors over a threshold (=365) of scoring model from the candidates. This step would reduce false positives in the candidates. To validate the effectiveness of our miRNA method, we evaluated the learned RCSG and the scoring model with test data. Here, we obtained satisfactory results, with high specificity (= 51/64) and proper sensitivity (= 51/82) using human miRNA precursors as a test data set.

1 Introduction

MicroRNAs (miRNAs) are a new class of ncRNA species, which are processed from long miRNA precursors by an enzyme Dicer [1]. The miRNAs participate in regulating gene expression directly by arresting the expression of specific mRNA, whereas other ncRNA participate in gene expression indirectly [2, 3, 4]. These miRNAs have been discovered by various experimental methods such as northern blot [5, 6], clone library [7], separation of microRNP, etc [8]. However, identifying miRNA by those experiments is considerably time-consuming and cost-expensive. Thus, we need a computational algorithm to efficiently predict miRNAs. The algorithms used for gene prediction are less efficient to predict miRNAs because miRNA sequences have a low similarity among sequences. Recently, some groups defined some statistical measures of miRNA precursors to predict miRNAs with respect to those of other species' miRNA precursors [6, 9]. However, these methods need the comparative analysis among miRNA precursors of evolutionarily similar species. If the miRNA precursors

C. Zhang, H.W. Guesgen, W.K. Yeap (Eds.): PRICAI 2004, LNAI 3157, pp. 732–741, 2004.

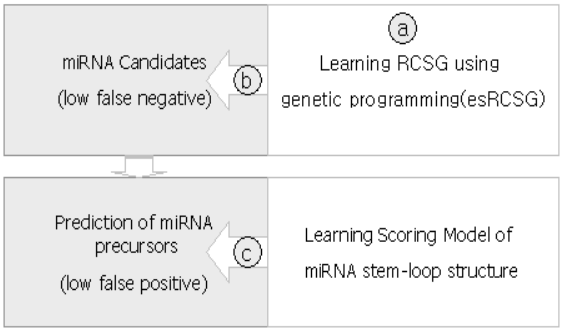

Fig. 1. The schematic overview of algorithm for predicting miRNAs. (a): Optimizing RCSG using genetic programming (b): Collecting miRNA candidates using the optimized RCSG (c): Selecting miRNA precursors from the candidates using the scoring scheme.

of one species have been not known, the method is impossible to predict putative miRNA precursors in the other similar species. Thus, it is essential to the general algorithm to identify putative miRNA only using the structure and sequence information of miRNA precursors. Also, the general algorithm is important to search the common structure and the conserved sequences. To do this, we applied genetic programming.

Genetic programming is an automated method to create working genetic programs, which are called individuals, from a high-level problem statement of a problem [10, 11]. These individuals are generally represented by tree structures. A genetic program consists of two elements, function symbol and terminal symbol. Function symbols appear as internal nodes. Terminal symbols are used to denote actions taken by the program. Genetic programming does this by genetically breeding a population of genetic programs using the principle of Darwinian natural selection and biologically inspired operations. Genetic programming uses crossover and mutation for transformation operators, which can endow variations to the genotype [10].

In this paper, we suggest a new method to detect putative miRNA precursors using a genetic programming and a scoring model of miRNA precursors. The method does not need the comparative analysis among similar species because it searches for the common-structure from miRNA precursors. To learn the common-structure, we define the RCSG (RNA Common-Structural Grammar), which is optimized by genetic programming.

2 Materials and Methods

2.1 Outline of Our Approach

In order to identify miRNA genes in genomes, we have developed an algorithm including a three-step (Figure 1). At first step, the genetic programming optimizes the RCSG according to a predefined fitness function (Figure 1a). The fitness function uses the specificity, the sensitivity and complexity of the structural grammars. At next step, with the optimized RCSG, we search for miRNA precursor candidates on ge-

nome sequences using the grammar-based RNA searching program [12] (Figure 1b). The precursor candidates should be filtered to diminish the false positive in the final results using the scoring model (Figure 1c).

```
begin                          /* Learning RCSG */
    t = 0                      /* generation */
    initialize P(t)            /* population */
    convert P(t)               /* tree to grammar*/
    evaluate P(t)
    while (not termination-condition) do
    begin
        S = S + above(P(t))
        t = t+1
        select P(t) from P(t-1)              /* selection */
        crossover-mutate P(t) except Best    /* genetic operators */
        convert P(t)
        evaluate P(t)                        /* fitness function */
        if (local search)
            while (not termination-condition) do
                j = j + 1
                P_j(t) = mutate P(t)
                if (evaluate P(t) < evaluate P_j(t) )
                        P(t) = P_j(t)
    end
end
```

Fig. 2. The pseudo-code of our genetic programming, which learns RCSG from a set of training data, known miRNA precursors.

2.2 Genetic Programming to Optimize RCSG

We have implemented a special genetic programming for the optimization of RCSG. The algorithm of the genetic programming can be illustrated by the pseudo-code as Figure 2. The procedures of the algorithm can be described as follows: (a) initialize the population with randomly generated trees; (b) convert all function trees into structural grammars; (c) calculate the fitness, specificity, sensitivity, and complexity for all grammars; (d) evaluate all structural grammars using the sensitivity, specificity and complexity against the positive and negative training set; (e) using ranking selection, select function trees that will generate offspring (next generation); (f) apply variations, such as mutation and crossover, with the selected function trees; (g) Iterate steps (b) through (f) for the user-defined number of generations.

2.2.1 Individual Representation

In order to convert structural grammars into function trees, we have defined the function f1, f2 and root as shown in Figure 3a. These functions can be formulated by some expression rules (Figure 3a). Therefore, using the expression rules, we can express the structural grammars (Figure 3c) as function trees (Figure 3b). We can use the function trees as individuals of genetic programming because one of the characteristics of genetic programming is that individuals are represented by tree structures.

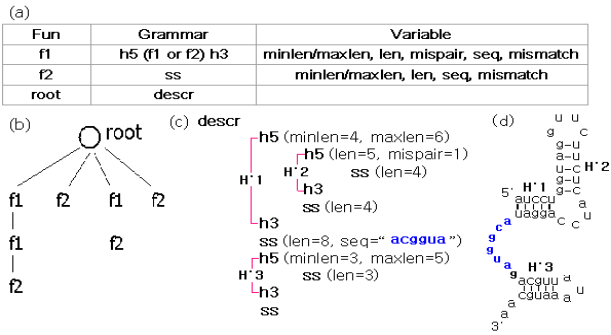

Fig. 3. (a): Function f1 generates recursively structural grammar, including one helix structure and either f1 or f2 as next deviation. Function f2 only represents ss (single strand), which means loop, bulge and single strand. Both f1 and f2 contain some variables, which measure structural information such as the length of helix (len), the number of pair (mispair) and mismatch, and sequence (seq). (b): A function tree to which can be converted into structural grammar (c): The child nodes of root in (b) conform to the first indentation of (c) and the nodes of second depth conform to the second indentation of (c). One helix that consists of the pair of h5-h3, h means helix, 5 and 3 mean 5' end and 3' end) (d): Secondary RNA structure is represented by structural grammar (c). H1, H2 and H3 in (c) and (d) are helix structures.

To avoid creation of invalid structural grammars, function trees have some constraints about the order of the function and the terminal node. First, f2 function should not appear consecutively in the same depth of the tree, contiguous f2 functions can be considered as only one. Second, f2 function can only appear as terminal node to terminate recursive generation of function tree. Finally, variables 'minlen' and 'maxlen' should always come in pair and should not coexist with variable 'len.'

2.2.2 Population Initialization

An initial population is randomly created with some constraints about function tree as described above. The initial population contains various function trees because there is no limitation in the number of nodes and the width of tree. That makes it possible to cover a wide range for searching start point. The broad coverage at start point is one of the major reasons the esRCSG is efficient for searching optimal solution.

2.2.3 Fitness Function

The fitness function (Equation 1) is defined by using specificity, sensitivity and the complexity that are defined at (Equation 4).

$$Fitness = spC * Specificity + stC * Sensitivity - Complexity \quad (1)$$

$$spC + stC = 1 \quad (2)$$

Two parameters, namely *spC* and *stC* were added as a way to regulate the effects of specificity and sensitivity. To normalize the fitness, the sum of *spC* and *stC* is always 1 (Equation 2). The parameters decide the trade-off between the specificity and the sensitivity on the fitness function.

The complexity, which is a negative effecter in fitness function, controls the growing of tree. Without the complexity term, the tree would not convert to the minimum size where the tree has best efficiency, but it would grow indefinitely during the evolution. To overcome that the over-sizing problem, we make $Comp_i^j$ include the node number and the depth of *j*th tree on *i*th generation (Equation 3). (Equation 4) describes the definition of *Complexity* of *j*th tree on *i*th generation.

$$Comp_i^j = TreeDepth_i^j \times 10 + NodeNum_i^j \tag{3}$$

$$Complexity_i^j = \frac{1}{(NS+PS)^2} \times \frac{Comp_i^j}{Comp_{i-1}^{best}} \tag{4}$$

where *Complexity* is normalized by square of the number of training data set (*NS* + *PS*). *NS* is the number of negative training data set and *PS* is the number of positive training data set. $Complexity_i^j$ depends on $Comp_{i-1}^{best}$, which is *Comp* of the best individual (tree) on (*i*-1)th generation. That dependency makes the trees have the minimum *Comp* as the progress of generation. Finally, the size of best tree on last generation is converged into minimum length as the principle of Occam's razor [13].

2.2.4 Variation

The variation operators are applied so that each descendent will have a different tree structure relative to the parents. The first operator to perturb the tree is the mutation. The mutation changes the value of the function variable by a random variable drawn from Poisson distribution. The crossover exchanges each sub-tree in two parent trees via single-point recombination to generate two new offspring trees. In the crossover, two parent function trees are selected at random from the population and then each single recombination point is selected at random from the each parent.

Table 1. Transition matrix.

	A	T	C	G
A	0.84			
T	-0.06	0.31		
C	-0.52	-0.24	1.40	
G	-0.47	-0.34	-0.93	1.28

2.3 Scoring Model to Predict miRNA Precursors

To construct the scoring scheme, we inspect the characteristics of the conserved sequences and structure over known miRNA precursors. Most miRNA precursors form long stem-loop structure of 70 ~100 nucleotides. Mature miRNAs are derived from the precursor transcript. Because the processing of the precursors is performed at specific region by Dicer enzyme, we believe that the conserved primary sequence or the conserved secondary structure exists in the stem-loop structure. Thus, we designed the scoring model with transition matrix, base paring score and IUPAC nucleotide ambiguity code (Equation 5).

$$Sm = \arg\max[\sum_{i=1}^{l}\sum_{j=1}^{n} \{S_{i,j} \cdot At\} + \sum_{i=1}^{l}\sum_{j=1}^{n} \{P_{i,j} \cdot Ap\}] \quad (5)$$

where are $S_{i,j}$ and $P_{i,j}$ a transition score and a pairing score respectively. *At* and *Ap* are constants that decide trade-off of transition score and paring score respectively. *n* is the number of all precursors and *l* is size of stem structure. The scoring model, *Sm*, is the paired sequences, which is represented with ambiguity codes and maximize the sum of the score.

We constructed the transition matrix through multiple structural alignments of miRNA precursors [14]. The transition matrix is described in Table 1.

3 Results

3.1 Dataset

For each step, training data set and test data set are described in Table 2. In order to extract extended stem-loop structures to be used as negative data set of scoring model in human genome, we predicted RNA secondary structures using RNAfold program (available at http://rna.tbi.univie.ac.at/ cgi-bin/RNAfold.cgi) for human chromosome 18 and 19. The stem-loop structures are selected under some criterions obtained through learning common structure of human miRNA precursors, which are sequence length (64 ~ 90 nucleotides), stem length (above 22 nucleotides), bulge size (under 15 nucleotides), loop size (3 ~ 20 nucleotides) and free energy (under -25 kcal).

Table 2. Data sets to train and test.

Steps	Class	Training number (Test number)	
Learning RCSG	Positive	50 (102)	http://www.sanger.ac.uk/software/
	Negative	200	Primary sequences, hairpins, RNA peudoknots, IRE, bulges and internal loops
Scoring model	Positive	85 (67)	http://www.sanger.ac.uk/software/
	Negative	1000	Stem-loop randomly extracted from Human chromosome 18 and 19

3.2 Learning RCSG of miRNA Precursor

The genetic programming succeeded in optimizing the RCSG of miRNA precursors (Figure 4a). The conditions and results of the experiment are described in Figure 4c. In this experiment, we tried a local search with the words of miRNA precursors. The optimal RCSG with the word 'gcaggga' allows one mismatch and has lower sensitivity than itself without the word. Figure 5 shows the plots of the fitness and the specificity of the best individual on each generation. Because using the elitism, the fitness readily increased or equaled according to growing generation.

(a)

```
descr (fitnss=0.93, specfty=0.96, sensty=0.71)
  h5(mispair=3)
       ss(minlen=5, maxlen=24)
       h5(len=5, mispair=1)
            h5(minlen=8, maxlen=14, seq='gcaggga', mismatch=1)
                 ss(minlen=5, maxlen=27)
                 h5(mispair=5)
                      ss(minlen=5, maxlen=15)
                  h3
            h3
       h3
  h3
```

(b)

```
descr (fitnss=0.74, specfty=0.84, sensty=0.84)
   h5(mispair=5)
        h5(mispair=1)
          ss(len=27, seq='gaguaaa', mismatch=0)
     h3
     ss(len=22)
   h3
```

(c)

Population size	30
Generation number	30
Iteration number for local search	30
spC	0.95
stC	0.05

Fig. 4. (a): The optimal RCSG of miRNA precursors. (c): The setting of parameters and the measures. (b): The RCSG is more general and more sensitive than the RCSG of (a) but less specific.

In order to evaluate the RCSGs for miRNA precursors, we performed the test with the test set consisting of 102 miRNA precursors (excluding training set) and 100 the negative data. In the results, we could measure sensitivity (= 0.71) with detecting 72 of 102 miRNA precursors and specificity (= 0.92) with 72 true positive of 78 positive candidates. The miRseeker of Lay group showed the validation results of the 75% (18/24) sensitivity and about 50% of specificity [1]. Our approach made a more specific identifier than miRseeker and could reduce the false positives. However, it seems that our strategy is less sensitivity than miRseeker [9]. If using alternative optimal RCSGs, such as Figure 4b, together, we think that it is enough to cover the low sensitivity.

3.3 Learning and Evaluation of Scoring Model

To reduce the false positives, we implemented the scoring model of stem-loop in miRNA precursor as described above. The scoring model was optimized with maximizing the Equation 5 (Figure 6).

Next, we tested the scoring model with a negative set and human miRNA precursors (Figure 7). Only thirteen of a thousand negative sets were bigger than the threshold (score = 365), the remaining was smaller. It means that our method shows high specificity. Also, 51 of 82 human miRNA genes were uncovered by our method and the sensitivity was 0.62, slightly higher than the sensitivity obtained by using miRseeker [9].

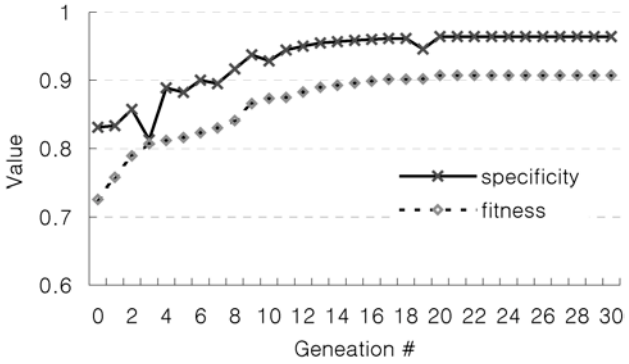

Fig. 5. The plot of the best RCSG for learning miRNA precursor.

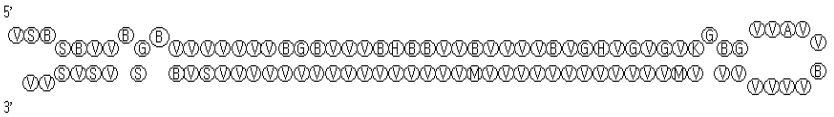

Fig. 6. The optimized scoring model of stem structure in miRNA precursor. This scoring model was learned in the direction of maximizing the score (Equation 5) for 69 training set (of human miRNA precursors).

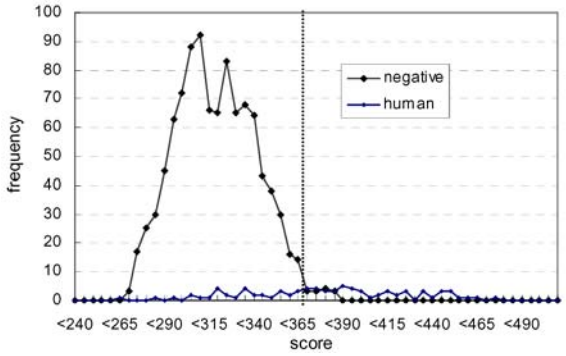

Fig. 7. The distribution of score of negative data and human as test data. The negative set was made of a thousand extended stem-loops extracted in human genome. The positive test sets consist of 82 sequences except training set. The dotted line is a score threshold.

Table 3. 5-fold Cross-validation of predicting miRNA precursor.

Accuracy (5-fold)	Average	Min	Max	S.D.
Sensitivity	0.63	0.56	0.70	0.06
Specificity	0.80	0.74	0.89	0.06

In general modeling situations, it is necessary to decide which models within a specific set are more compatible with all given data. Cross-validation is one established way for accomplishing this. We describe the result for 5-fold cross-validation [15] as shown in Table 3. The 152 miRNAs are randomly divided into 5 groups.

4 Discussions

In this paper, we suggested an effective approach to search for a RCSG with miRNA sequences based on the genetic programming. Also, we introduced a new approach to predict putative miRNA precursors via the RCSG and the scoring model. When applying the human miRNA precursors, we could find the distinctive RCSGs. The identified RCSGs reflected the common structures of miRNA precursors.

We can derive two contributions from our new approach. The first contribution is that we have proven the possibility to learn common-structural grammar from structurally unknown sequences through genetic programming. We believe that our approach can be applied for various applications such as RNA similarity search and putative RNA identification. The second contribution is that we have suggested a new approach to predict putative miRNA. Our approach is a sufficient method to minimize false positives because our method applies the common structure and the information of evolutionarily conserved sequence and structure. We believe that our approach can help many biologists identify putative miRNAs.

For further works, we will consider detail grammars, such as if statements, to increase control over RCSG learning process and will apply Hidden Markov models (HMM) instead of the scoring model to make stochastic learning model. Also, because the efficiency of the learning depends on how we constitute the negative training set, it is important to associate apt sequences having various secondary structures for the negative training sequence set. We are trying to develop a method to make more optimal negative training set.

Acknowledgments

This research was supported by BK21-IT, Korean System Biology Research Grant (M10309000002 -03B5000-00110) and the NRL (M10203000095-03J0000-04510).

References

1. Huttenhofer A., Brosius J. and Bachellerie JP. RNomics: identification and function of small, non-messenger RNAs. *Current Opinion in Chemical Biology*, 6:835-843, 2002.
2. Ambros V. Tiny regulators with great potential. *Cell*, 107:823-826, 2001.
3. Gottesman S. *Genes & Development*, 16:2829-2842, 2002.
4. Zamore P. D. Ancient Pathways programmed by small RNAs. *Science*, 296:1265-1269, 2002.
5. Lagos-Quintana M., Rauhut R., Lendeckel W. and Tuschl T. Identification of novel genes coding for small expressed RNAs. *Science*, 294:853-858, 2001.
6. Lim L. P., Glasner M. E., Yekta S., Burge C. B., and Bartel D. P. Vertebrate microRNA genes. *Science*, 299:1540, 2003.
7. Lagos-Quintana M., Rauhut R., Meyer J., Borkhardt A. and Tuschl T. New microRNAs form mouse and human. *RNA*, 9:175-179, 2003.
8. Dostie J., Mourelatos Z., Yang M., Sharma A., and Dreyfuss G. Numerous microRNPs in neuronal cell containing novel microRNA. *RNA*, 9:180-186, 2003.

9. Lai E.C., Tomancak P., Williams R.W. and Rubin G. M. Computational identification of Drosophila microRNA genes. *Genome Biology*, 4:R42. 2003.
10. Koza J. R. Genetic programming: On the programming of computers by means of natural selection. MIT Press. 1992.
11. Angeline P.J. and Kinnear K.E. Advances in Genetic Programming 2. Jr MIT Press. 1996.
12. Thomas J. Macke, David J. Ecker, Robin R. Gutell, Daniel Gautheret, David A. Case and Rangarajan Sampath. RNAmotif, an RNA secondary structure definition and search algorithms. *Nucleic Acids Research*, 29:4724-4735, 2001.
13. Zhang B.-T., Ohm P., and Mühlenbein H. Evolutionary neural trees for modeling and predicting complex systems. *Engineering Applications of Artificial Intelligence*, 10:473-483, 1997.
14. Siebert S. and Backofen R. MARNA: A Server for Multiple Alignment of RNAs. *In Proceedings of the German Conference on Bioinformatics*, 135-140, 2003.
15. Weiss S.M. and Kapouleas I. An empirical comparison of pattern recognition neural nets and machine learning classification methods. *In Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Detroit, Mich: IJCA1 234–237, 1989.