A Novel Algorithm for DNA Multiple Sequence Alignment Based on the Sliding Window and the Keyword Tree

Yong Sun, Zili Zhang, and Jun Wang

Abstract—**Multiple sequence alignment (MSA) is a difficult yet important problem for bioinformatics research. In most cases, large-scale biological sequence data with high similarity have to be analyzed. Center star method is always used to deal with lots of long sequences. However, square time complexity is a bottleneck for large data. In this paper, a novel method for the MSA problem is proposed, which employs the keyword tree and the sliding window to match a set of substrings and the rest regions are aligned by dynamic programming. The method provides the dynamic adaptive mechanism for the sliding window size and step length. The self-adaptive parameters play a extremely important part for improving the performance of the method. Experimental results show that the proposed method is computational efficient and can obtain good performance.**

***Index Terms*—Multiple sequence alignment, center star method, keyword tree, sliding window.**

I. Introduction

The sequence alignment, especially MSA, has a great significance for the discoveries and studies of the genetic functions, structures and evolution processes of the biological sequences [1]. With the help of the sequence alignment methods, biologists can find the conserved sequence patterns in the evolution procedure and reveal the ancestral relationships among different organisms. Especially, rapid development of computational molecular biology demands well-efficient sequence aligning algorithm crucially [2], [3]. Nowadays there have been a variety of algorithms proposed, e.g. Intelligent optimization, the probability model and the parallel mechanism. Combinatorial optimization algorithm is one of the most effective ways to solve the MSA problem, the idea of which is converting a MSA problem to many pairwise sequence alignment problems. This method can be divided into two types according to the transformation strategy, one is the tree alignment based methods [4] and the other is the star alignment based methods [5], [6]. At the same time, the progressive theory is introduced in these methods and makes these methods more efficient. This greedy heuristic assembly algorithm involves estimating a guide tree (rooted binary tree), and then incorporating the sequences into MSA with a pairwise alignment algorithm while following the tree topology. The progressive algorithm is often embedded in an iterative loop where the guide tree and MSA are reestimated until convergence [7], [8]. However, the time complexity of the simple star alignment based methods is not good enough. These methods also cannot adapt to align the large-scale sequence data. Thus, we need to find a more reasonable method to solve this problem.

Through the analysis of the genetic sequence data, we find that most sequences have high similarity that can be employed in bioinformatics research. For example, for reconstructing evolutionary trees and comparing haplotype sequences, it needs to align many DNA sequences with high similarity [9], The high similarity means that many substrings on the r-length regions are exact match between sequences. Thus, the proposed method mainly concentrates on searching mismatching r-length regions. This method can be divided into two parts: the reference sequence determination and the sequence alignment. The keyword tree and the sliding window are constructed to quickly search the mismatching substrings between two sequences and ensure the accuracy of the searching.

Manuscript received December 9, 2012; revised February 25, 2013. This work is supported by the Natural Science Foundation of China under Grant No. 61101234, the Doctoral Funds of Southwest University under Grant No.SWU110063.

Yong Sun, Zili Zhang, and Jun Wang are with School of Computer and Information Science, Southwest University, Chongqing, 400715, China (e-mail: likesy999@yahoo.com.cn, zhangzl@swu.edu.cn, kingjun@swu.edu.cn).

Zili Zhang is with School of IT, Deakin University, Australia.

II. Definitions and Conceptes

In the proposed method, several basic and important theories and methods are used to find the appropriate reference sequence and obtain the final alignments, such as the keyword tree, the sliding window, etc. Thus, in this section, we will give some important definitions and related concepts.

A. *Sequence Alignment*

A sequence is actually a string over an alphabet collection. For DNA sequences, the alphabet collection contains four letters A, C, G and T, representing four distinct nucleotides respectively. Given two or more strings, the aim of the sequence alignment is making them the highest similarity based on SP (sum-of-pairs) [6]. To reach this aim, the alignment algorithm often adds a space in the strings. A space is viewed as a letter (A, G, C or T) and is denoted " - " throughout this paper. Two opposing identical letters form a match and two opposing non identical letters form a mismatch.

Sequence alignment algorithm can obtain the optimal alignment of two or more strings according to a given scoring function. That is, the results of sequence alignment can reflect the relationship of sequence similarity and their biological characteristics. Take ATACTAGA and AACTTGGA for an example, the following is the optimal alignment after arrangement.

ATACTAG -A
A-ACTTGGA

The whole procedure can be viewed as "adding space" in order to make sequences have the highest similarity. The sequence alignment can be divided into the pairwise and the multiple sequence alignment based on the number of comparing sequences at the same time. Besides, on the basis of the scope of the comparison, it also can be divided into the global alignment and the local alignment. A global MSA algorithm is defined here as one that tries to align the full length sequences from one end to the other. Once the global alignment has been constructed, other methods are often used to identify the more conserved or reliable regions within the alignment. A local algorithm attempts to identify subsequences sharing high similarity. The unreliable or low similarity regions are then either excluded from the alignment, or differentiated, for example, by the use of upper/lower case characters.

Currently, the dynamic programming is the main method for pairwise sequence alignment [10]. But it is no good way to solve the multiple alignment problems on large scale sequences. The time complexity of dynamic programming is too high to extend it to multiple sequence alignment. The method requires the computation of all $\binom{n}{2}$(n is the number of strings) optimal pairwise alignments. For large n and long strings, this may involve a great deal of computation.

B. Keyword Tree

A keyword tree is built by a collection of several short strings [11]

$$P = \{P_1, P_2, \cdots, P_z\}$$

$(z \in N, z > 1)$, P_i(i=1,2, $\cdots$,z) is a short string or a substring from one given long sequence. The root of the keyword tree is K. Every edge of the tree represents a letter of P_i. Different edges, which are separated from the same vertex, represent diverse letters of the strings with same prefix. Each P_i relates a path from the root K to a leaf node. In other words, a leaf node can also represent a short string P_i. Fig. 1 gives an example of a keyword tree of

P_i={ATACTA,CTATGC,CATG,TACT}

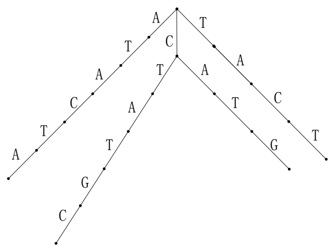

Fig. 1. Keyword tree of P.

The keyword tree K is utilized to search another given long string T to find the elements of P. The searching approach is based on the sliding window theory.

C. Sliding Window

Sliding window can be seen as a limited circulated internal memory which is based on array [12]. It is one of the important data flow processing models. It stores the data from some region of the data flow. The data move in order according to the given step length in array. Old data leave out of the window as the new data enter into the window. Fig. 2 gives a simple model of sliding windows. The window size is w and the sliding step length is s. s equals to the length of the window movement each time. The black-blocks represent the current place of window in Fig. 2.

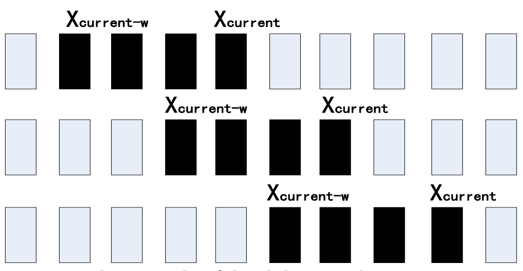

Fig. 2. The example of the sliding window (w=4, s=2).

In the method, the window size is constant when the algorithm is doing one specific comparison. For different biological sequence data, the algorithm is self-adaptive and chooses the appropriate values of the parameters. The sliding window moves on and compares the sub-strings in the windows with the other ones. The matching and mismatching pairs are marked. By calculating the number of matching sub-strings between the given sequences, the sequence with the largest number can be found and is set as the reference sequence at last. Then, it can be used in the following sequence aligning process.

III. Method

To describe the method, we firstly need to know how to make use of the keyword tree and the sliding window to find the appropriate reference sequence and speed up the sequence alignment process. Here we need to pay attention to it that the square of sequence number and the square of the average length of sequences are proportional with the time overhead of the center star method, regardless of similarity.

The square time overhead in the center star method is mainly due to the time overhead of dynamic programming. However, if the similarity of the sequences is high, the exact matching substrings have a large proportion, that is, only a few mismatching r-length regions will need lots of the running time. And the using time of the string matching and the sequence length is a linear relationship. Then we just need to align the remaining mismatching r-length regions with dynamic programming method [13]. It can take full advantage of the high similarity and the aligning time will be greatly reduced.

Given the sequence set $\{p_1, p_2, \cdots, p_n\}$, Each p_i (i=1,2, $\cdots$,n) is divided into k segments $\{p_{i1}, p_{i2}, \cdots p_{ik}\}$ with equal length. The length of each segment is r. If the length of the last segment is not enough, it can be ignored. A keyword tree T_i is built based on the segment set of p_i. L is the average sequence length of all p_i(i=1,2, $\cdots$,n). Due to the high similarity of the given sequences, the difference among the sequence lengths is so small that it will not affect the selection of r.

For each p_j ($j \neq i$), the algorithm searches p_j to mark the matching substring between p_j and the keyword tree T_i using the sliding window. The window size is also set as r. For each substring in the window, which is denoted by p, will compare following the branch of T_i and search for the matching ones. The search stops at once when it encounters a mismatching character. If all characters in p match a branch of T_i, the similarity degree of p_i add one, And the algorithm also notes down the serial number of the exact matching substring in T_i and the location of p in p_i.

When a comparison in one window is finished, the sliding window moves forward in a step length s. The same matching action is circularly done. The selected step length s is corresponding to the number and the length of the given aligned sequences. We will discuss it in detail later. When the window slides to the end of p_j, the algorithm records the number n_{ij} of all exact matching substrings. After all p_j compared with p_i, the algorithm calculates the total similarity of p_i to the other sequences. This procedure is circularly done on each p_i. That means the algorithm builds keyword tree for each sequence and calculates the similarities of them. The algorithm selects the one which has the biggest Ni as the reference sequence. We have found out the reference sequence p_c through the algorithm described above. Then p_c will align with p_j sequentially, $j \in \{1,2,\cdots,n\}$, $j \neq c$. The algorithm has found out all mismatching r-length regions through the recorded serial number and location in p_c and p_j. Then, the dynamic programming is employed to align all sequence pairs (p_c, p_j).

Only the mismatching r-length regions in the sequence pairs are aligned. The aligning algorithm needs to record the location of inserted spaces in p_c and p_j, which are donated by S_{ci} and S_i. When p_c has aligned with all the other sequences, the algorithm can obtain a space location set S_c, which should insert spaces finally. Then, S_c is compared with S_{ci} respectively, and the algorithm can find the new positions that need to insert space in S_i. That means it will insert new space into p_j based on S_i to make it have the highest similarity. This step runs circularly and finally obtains the result of MSA. The detail procedure of the whole aligning algorithm can be shown as follows [14].

For the alignment algorithm of the paper, supposed the length of sequences $\{p_1, p_2, \cdots, p_n\}$ are all m. The time complexity of building a keyword tree is O(m). Besides, the time is O(nmx) which is spent in searching in other strings and computing n_{ij}. x is relative with the sliding step length. When s is suitable for the data set, x will be less than 1. On the contrary, x will be more than 1.

Thus, for $i \in \{1,2,\cdots,n\}$, the total time complexity is O(n^2mx) after computing all n_{ij} and finding the reference sequence. After the reference sequence selection, it needs to n-1 k-band pairwise alignments. The time is O(knmx). k is the difference between two sequences. When it is the high similarity among sequences, k is far less than m (k << m). Therefore, compared with the reference sequence selection, the time overhead of the n-1 pairwise alignments can be ignored. The expected time complexity of the algorithm is O(n^2mx). When x < 1, the running time is lower than O(n^2m) of OA. OA is the original algorithm (see [15]). Generally speaking, n is far less than m (n << m). nm is the sum of sequences' lengths. So it is considered that it is a linear relationship between the time overhead and the sum of the input sequence lengths.

```
Algorithm: Sliding window algorithm on MSA
Input： $\{p_1, p_2, \cdots, p_n\}$
Output：$\{P_1, P_2, \cdots, P_n\}$
Step 1.   for $i \in \{1,2,\ldots,n\}$
Step 2.    r= $\sqrt{L}$
           //L:the average length of sequences
Step 3.    Dividing $p_i$ into k substrings,building the
           keyword tree $Ti$ for the substring set
Step 4.    for $j \in \{1,2,\ldots,n\}$, and $j \neq i$
              $n_{ij}=\{0,0,\ldots,0\}$
Step 5.       Searching $T_i$ in $p_j$
Step 6.       If p in $p_j$
               //p:substring within the sliding window
Step 7.         1 for the corresponding position of $n_{ij}$ and
                record the location
Step 8.        End if
Step 9.      End for
Step 10.   End for
Step 11.   for $i \in \{1,2,\ldots,n\}$
Step 12.       $N_i = \sum_{j=1, j \neq i}^{n} n_{ij}$
                //calculate the similarity of $p_i$
Step 13.   End for
Step 14.   The biggest one of $N_i$ is set as the reference
            sequence $p_c$
Step 15.      for $j \in \{1,2,\ldots,n\}$, and $j \neq c$
Step 16.      Aligning parts which is 0 in $n_{ij}$ and recording
               positions of spaces
Step 17.      End for
Step 18.   Inserting spaces according to aggregating
           positions
Step 19.   Output $\{P_1, P_2, \cdots, P_n\}$
```

IV. Experiment and Disscussion

There are two main evaluation criteria for MSA: the running time and the sensitivity. Naturally, a good aligning algorithm has low running time and excellent sensitivity (low SP here). However, it is usually very difficult to keep a balance between two criteria. Researchers found that the algorithm can reach a relatively balance on running time and sensitivity by using some adaptive categories on some kinds of favorable parameters, such as the similarity. The step length of the sliding window is such crucial parameter for reaching this balance. If the step length sets too short, the algorithm will waste a lot of time to search and compare the substrings. Especially, when the dataset has too many sequences and all sequences are very long, the short step length will spend the very long running time on the reference sequence determination of the algorithm. In contrast, if the step length is too long, there are lots of exact matching r-length regions missed. That will make the relative low sensitivity and the high running time may be spent in aligning step, since the chosen mismatching r-length regions may be inaccuracy.

A. *Experimental Results*

In this section, we will show the experimental results and analyze the performance of the algorithm. The algorithm is implemented by a single CPU of a Pentium(R) Dual-core E5700 3.00GHz and its RAM is 2.00GB.

The selection of the step length has a great relationship with the aligning data size and the sequence length. Through a great number of training experiments, we have found out a set of the appropriate step lengths. Firstly, we choose three representative experimental data to show the significance of these parameters and its advantage. All three data are mitochondrial DNA sequences: HNsq Thin young males (HN); JDsq Type 2 diabetes patients with angiopathy (JD); KAsq Alzheimer's disease patients (KA). Each of these sequences has about 16, 570 bp and the size of each data is about 96. Because they belong to the different individuals of the same species, they have very high similarity naturally.

TABLE I: THE PERFORMANCE OF THREE DATA ON OA AND MA

Dataset		OA	MA
HN	Running time	52728	39549
HN	SP	185258	186268
JD	Running time	55387	39250
JD	SP	196080	195945
KA	Running time	59266	39120
KA	SP	178868	178868

Table I shows the results of the comparison between MA and OA. MA is the modified algorithm in this paper. Experimental units of the running time are all millisecond (ms) in this paper.

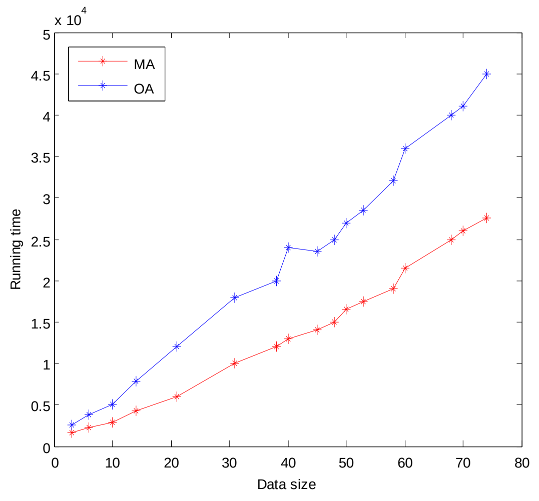

Fig. 3. The running time of OA and MA on different data size.

From Table I, it shows that MA greatly reduces the running time and the accuracy is slightly improved or remains the same too. Here the algorithm selects 2 as its step length by adaptive selection mechanism on the dataset size.

In order to find a suitable sliding step, we must consider a variety of situations. Firstly, we do some experiments to verify the influence of the change of the data size to the results, which are shown in Fig. 3, where the sliding step is 2. In the experiments, all the sequence length is about 16,570 bp. We can find the running time of MA is greatly lower than the running time of OA.

When the sliding step is 1, we find that its performance is very unstable through many experiments. The running time would be very long, even can't run. It leads directly to the badly operating results. In addition, we can't only know the advantage of MA through the change of the data size, but also we see the influence of sequence length on the sliding step. Similarly, for the long sequences, about 16,570 bp, in Fig. 4, the data size does not change, but the sequence length changed. All the data size is 8.

The sliding step of Fig. 4 is 1. When the sliding step is 2, its performance is very unstable. The running time would be very long, even can't run. In Table II, the sliding step is 1 and 2(only the representative portion shown). OA and MA are the representative of their respective running time. * means that the algorithm can't run or runs for a long time but still no results.

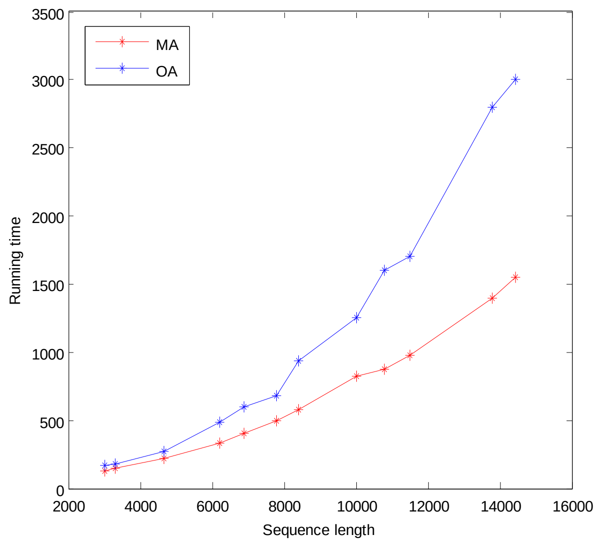

Fig. 4. The running time of OA and MA on different sequence length.

TABLE II: THE PERFORMANCE OF THE DATA ON OA AND MA

Sequence length	3601	3336	4625	6924	10789	14362
OA	160	173	263	650	1632	3021
MA(s=1)	130	151	233	430	960	1577
MA(s=2)	135	1615	2862	465	*	*

TABLE III: THE PERFORMANCE OF DENGO ON OA AND MA

	OA	MA	MA
Step length	-	1	2
Running time	370	320	435
SP	8504	8504	8504

TABLE IV: THE PERFORMANCE OF THE DATA ON OA AND MA

	Step length	Data size	Running time	SP
OA	-	2	2441	8132
MA	1	2	1424	8147
MA	2	2	1616	8147

According to the two Figures above, when the data size or the sequence length is small, the running time of OA and MA is fairly close. With the increase of data size or sequence length, the running time of MA increases slower and smooth, and the increase of OA is faster and steep, the gap between them becomes larger and more evident slowly. This shows sequence length and the data size affects the performance of the algorithm. At the same time, the experiments also prove that the sequence length and the data size affect the sliding step, i.e. the efficiency of this algorithm. At the same time, the advantage of MA is apparent for the longer sequence and bigger data size.

For the data which has relative short and small sequences, the algorithm adaptively chooses the short step length and also proves its superiority. This can be proved by the experiment (the results in Table III) on the Dengo virus strain data [16], which has only 17 sequences and the sequence length is about 1485bp. When the step length is 1, the running time is lower than OA's, but if the step length is 2, the running time is even higher than OA's.

All these experiments show an appropriate step length is the critical for the algorithm. And the experimental results prove that our method can obtain reasonable alignments in shorter time.

B. Discussion

From Table I, we know that the running time of MA is about 70% of OA. SP of MA is an increase of 2% for HN, decrease of 0.1% for JD and no change for KA. Because the results of the center star method have been the approximate optimal solution, it is reasonable for the slightly higher SP of MA on some special data. Therefore, it does not affect the conclusion and analysis of biological experiments. And MA is lower than others in running time.

The influence of the data size to the parameter choosing is a gradual change process and not a sudden change point. Through the experiments, like Fig. 4 and Table II, we find the sequence length can directly affect the results of the algorithm. In the experiment shown in Table IV, we use a dataset with 2 sequences. Each sequence has about 20,000bp. The running times of MA are smaller than OA's on each value of the step length s. There are the same SP and the near running time when s=1 and s=2. There are the similar running time, mainly because the data size is so small, just 2. This sample is different from the Dengo virus strain data whose running time is poor for s=2.

For the sequences with high similarity, especially the large-scale data, the data size and the sequence length of which are numerous and long respectively. And the higher the similarity, the more it can search the matching r-length regions with the sliding window. In addition, s=2 can accelerate the searching and the matching r-length regions we may ignore are less. In other words, the less the searching mismatching r-length regions are, the less we use the dynamic programming to align. Then, it saves time overhead.

REFERENCES

[1] A. Luytynoja and N. Goldman, "Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis," *Science*, vol. 320, no. 5883, pp. 1632-1635, 2008.

[2] R. C. Edgar and S. Batzoglou, "Multiple sequence alignment," *Structural Biology*, vol. 16, no. 3, pp. 368-373, 2006.

[3] D. Gusfield, "Algorithms on strings, trees and sequences: Computer Science and Computational Biology," *Cambridge England*, Cambridge University Press, 1997, pp. 505-523.

[4] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence alignment of two proteins," *Journal of Molecular Biology*, vol. 48, pp. 443-453, 1970.

[5] L. Wang and D. Gusfield, "Improved approximation algorithms for tree alignment," *Journal of Algorithms*, vol. 25, no. 2, pp. 255-273, 1997.

[6] S. F. Altschul and D. J. Lipman, "Trees, stars and multiple biological sequence alignment," *SIAM Journal on Applied Mathematics*, vol. 49, no. 1, pp. 197-209, 1989.

[7] D. Gusfield, "Efficient methods for muliple sequence alignment with guaranteed error bounds," *Bulletin of Mathematical Biology*, vol. 55, no. 1, pp. 141-154, 1993.

[8] A. R. Subramanian, M. Kaufmann, and B. Morgenstern, "DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment," *Algorithms for Molecular Biology*, vol. 3, no. 6, 2008.

[9] S. Batzoglou, "The many faces of sequence alignment," *Briefings in Bioinformatics*, vol. 6, pp. 1, pp. 6-22, 2005.

[10] J. Z. Li, D. D. Zhang, and S. Takasaki, "Mitochondrial SNPs associated with Japanese centenarians, Alzheimer's patients, and Parkinson's patients," *Computational Biology and Chemistry*, vol. 32, no. 5, pp. 332-337, 2008.

[11] C. Chica, A. Labarga, C. M. Gould, R. Lopez, and T. J. Gibson, "A tree-based conservation scoring method for short linear motifs in multiple alignments of protein sequences," *BMC Bioinformatics*, vol. 9, no. 229, pp. 1471-2105, 2008.

[12] J. Z. Li and D. D. Zhang, "Algorithms for dynamically adjusting the sizes of sliding windows," *Journal of Software*, vol. 15, no. 12, 2004.

[13] Y. Tang and M. Wang, "Fast sequence alignment algorithm based on dynamic programming," *Journal of Biomathematics*, vol. 20, no. 2, pp. 207-212, 2005.

[14] J. Wang, M. Z. Guo, and Q. Zou, "An mtSNPs based method for disease population discrimination," *China National Computer Conference*, Tianjin, China, 2009, pp. 739-746.

[15] Q. Zhou and M. Z. Guo, "An algorithm for DNA multiple sequence alignment based on center star method and keyword tree," *Chinese Journal of Electronics*, vol. 37, no. 8, pp. 1746-1750, 2009.

[16] A. Rambaut, "Estimating the rate of molecular evolution: incorporating non-contemporaneous sequences into maximum likelihood phylogenies," *Bioinformatics*, vol. 16, no. 4, pp. 395-399, 2000.

Yong Sun was born in Sichuan, July 25, 1986. He is a postgraduate student in the School of Computer and Information Science, Southwest University, Chongqing, China. Current research interests are machine learning and its application in the multiple sequence alignment.

Zili Zhang was born in Chongqing, Dec. 29, 1964. He is a professor at Southwest University, Chongqing, China, and a senior lecturer at Deakin University, Australia. He received his BSc from Sichuan University, MEng from Harbin Institute of Technology, and PhD from Deakin University, all in computing. He authored or co-authored more than 100 refereed papers in international journals or conference proceedings, 1 monograph, and 4 textbooks. His research interests include bio-inspired artificial intelligence, agent-based computing, big data analysis, and agent-data mining interaction and integration. Contact him at zzhang@deakin.edu.au or zhangzl@swu.edu.cn.

Jun Wang was born in Chongqing, March 5th, 1983. She received her PhD in Artificial Intellegence from the School of Computer Science, Harbin Institute of Technology, China, in 2010. She now works as an Associate professor in the School of Computer and Information Science, Southwest University. She has published more than 13 papers in bioinformatics field. Current research interests are machine learning, data mining and their applications in bioinformatics.