American Journal of Biochemistry and Biotechnology

Original Research Paper

A Selection-based MCL Clustering Algorithm for Motif Discovery

[1]Chunxiao Sun, [2*]Zhiyong Zhang, [2]Jinglei Tang and [1]Shuai Liu

[1]College of Science, Northwest A&F University, Yangling, Shaanxi, 712100, China
[2]College of information Engineering, Northwest A&F University, Yangling, Shaanxi, 712100, China

Article history
Received: 05-09-2018
Revised: 03-12-2018
Accepted: 28-12-2018

Corresponding Author:
Zhiyong Zhang
College of Information Engineering, Northwest A&F University, Yangling, Shaanxi, 712100, China
Email: caich1980@sina.com

Abstract: As motif discovery plays an important role in the understanding of the relationship of gene regulation, this paper puts forward a selection-based MCL clustering refinement algorithm (SMCLR) aiming at solving the planted (*l*, *d*) motif search (PMS) problem. Firstly, we divide the DNA dataset into different subsets through selection of reference sequence and screen parts of eligible subsets by setting thresholds under selection project. Then MCL clustering algorithm is used for refinement. The experiment resulted on simulation data shows that SMCLR algorithm has higher prediction accuracy in a reasonable time than these existing motif discovery algorithms like Project, MEME, MCL-WMR and VINE. Moreover, the experiment resulted on real biological data demonstrates the effectiveness of SMCLR algorithm.

Keywords: Motif Discovery, MCL Clustering, Gene Regulation, Bioinformatics

Introduction

Motif discovery acts a pivotal part in understanding gene finding and gene regulation relationship, also is one of the most challenging problems in molecular biology and bioinformatics (Jones and Pevzner, 2004). Although many motif discovery algorithms have been proposed, Das and other scientists (Das and Dai, 2007) argue that DNA motif discovery would be a prevalent challenge in recent years.

According to the combination method in algorithm design, the existing motif discovery algorithms can be divided into exact algorithms and approximate algorithms. The former (Marsan and Sagot, 2000; Eskin and Pevzner, 2002; Evans and Smith, 2003; Pisanti *et al*., 2006; Pavesi *et al*., 2001; 2004; Davila *et al*., 2007; Dinh *et al*., 2011; Bandyopadhyay *et al*., 2012; Dinh *et al*., 2012; Ho *et al*., 2009; Rajasekaran *et al*., 2005; Tanaka, 2014) usually uses a consensus sequence (Hertz and Hartzell, 1990) to represent motif and finds the optimal solution by traversing the entire solution space. At present, the major method for exact algorithm research means to validate all *l*-mers (an l-length string) on the O(4-l) search space to output all the *l*-mers with the motif properties, which are the *l*-mers with a Hamming distance of d between motif in each sequence. Among these algorithms, a type of exact algorithms construct the input sequences into the suffix tree to expand the efficient pattern search, such as SMILE (Marsan and Sagot, 2000), MITRA (Eskin and Pevzner, 2002), CONSUS (Evans and Smith, 2003), RISOTTO (Pisanti *et al*., 2006) and Weeder (Pavesi *et al*., 2001; 2004). These algorithms first establish the suffix tree of the input sequences and then validate all *l*-mers patterns. They would quickly resolve PMS problem in short motif with suffix trees, but time performance is keenly sensitive to long motif. Another type of exact algorithms adopt the pattern-driven method, which are to establish candidate motif set according to input sequence and to verify the validity of each candidate motif, such as PMSPrune (Davila *et al*., 2007), PMS5 (Dinh *et al*., 2011), PMS6 (Bandyopadhyay *et al*., 2012), qPMS7 (Dinh *et al*., 2012) and iTriplet (Ho *et al*., 2009). These algorithms first choose a reference sequence and then limit the search space with the combination of other sequences, reducing the total number of validated patterns.

The approximate algorithm (Bailey *et al*., 2006; Boucher, 2007; Neuwald *et al*., 1995; Lawrence *et al*., 1993; Buhler and Tompa, 2002; Fratkin *et al*., 2006; Boucher *et al*., 2007) usually uses the Position Weight Matrix (PWM) (Stormo *et al*., 1982) to represent motif and report the position weight matrix with the highest score by updating the position weight matrix iteratively. For example, typical motif discovery algorithms are MEME (Bailey *et al*., 2006), Gibbs Sampling (Neuwald *et al*., 1995; Lawrence *et al*., 1993) and the extended related algorithms (Buhler and Tompa, 2002). MEME expands the EM algorithm, improves the capability of the EM algorithm in local

search and gets around the shortcoming that the EM algorithm may converge to local optimum. Gibbs sampling first randomly selects the starting position in each sequence to generate the initial state and then executes the two-step iteration of updating and sampling. In addition to statistical methods, other methods based on clustering algorithm are proposed such as Motif-cut (Fratkin *et al*., 2006) and MCL-WML (Boucher *et al*., 2007). Motif-cut searches for the maximum density subgraph by looking for the maximum flow and the minimum cut in graphs to find motif. MCL-WMR adopts the Markov Cluster (MCL) clustering algorithm to search for the largest cluster in graph.

It is very difficult to improve the performance of algorithms only by relying on exhaustive or local search strategies, but ideal result may be achieved by making high quality refinement at the basis of effective dimensionality reduction. Therefore, a selection-based MCL clustering refinement algorithm is proposed in this paper. Firstly, the *l*-mers in the input sequences are selected into a series of buckets by using the dimension reduction strategy and a number of qualified buckets are formed, each of which corresponds to a candidate motif set. Then, the MCL clustering refinement is used for each qualified bucket to obtain motif. For most planted (*l*, *d*) motif search (PMS) problem (Buhler and Tompa, 2002), the prediction accuracy of the SMCLR algorithm is improved to some extent and the experiment results on both simulation dataset and real biological dataset show that the SMCLR algorithm not only discovers the motifs as consistently as the published ones but also quite efficiently.

Materials and Method

Background

Problem Definition

Given a set of DNA sequences with a length of n and a nonnegative integer l, each input sequence contains a string of length l, which differs to the same string up to d positions. Planted (*l*, *d*) motif search (PMS) problem aims to find out the same string and its mutated string in each sequence. The formal definition of the problem is given below:

PMS problem: $S = \{S_1, S_2,\ldots,S_t\}$, t sets of DNA sequences with a length of n in alphabet {A，C，G，T} and two nonnegative integers l and d ($0 \le d < l < n$), are known. Planted (l, d) motif search is to find an l-mer M (a string with a length of l), so that each sequence S_i contains an l-mer M_i, mutated from M at up to d positions. The l-mer M is called an (l, d) motif and the l-mer M_i is called a motif instance of M.

A motif is usually represented by a consensus sequence or a Position Weight Matrix (PWM). A consensus sequence is a character string with a length of l and each character is the most frequently exposed one in the corresponding column. A position weight matrix is a $4\times l$ matrix, in which each element represents the frequency of the corresponding character in the corresponding column. Figure 1 depicts the construction process of consensus sequence and position weight matrix in the simulation data.

We use the Information Content (IC) (Liu *et al*., 2001) to evaluate the conservation of a motif.

$$IC = \sum_{w=1}^{l} \sum_{k\in\{A,C,G,T\}} p_{k,w} \log \frac{p_{k,w}}{p_{k,0}}$$

where, $p_{k,w}$ is the probability of base k at the position w of the motif. $p_{k,\,0}$ is the probability of base k in the background sequence. Information content is a method of measuring the degree of conservatism of the motif using the background information of DNA sequence, in which the higher the value of IC, the greater the conservatism of a motif.

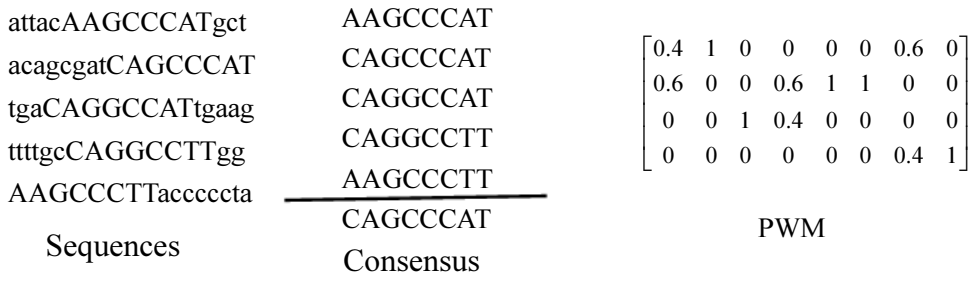


Fig. 1: An example of the construction process of consensus sequence and position weight matrix

MCL Cluster (Markov Cluster)

Markov Cluster (MCL) clustering (van Dongen, 2000) algorithm is a kind of graph clustering algorithm with a relatively good clustering performance at present. Based on transition probability matrix without presetting the number of clusters, the MCL clustering algorithm simulates random walks upon the underlying graph through performing expansion operator and inflation operator on a probability matrix alternately until the matrix converges. The flowchart of the MCL clustering algorithm is shown in Fig. 2:

(1) Data input: adjacent matrix A with self-loops, expand parameter e and inflation parameter r

(2) Expand: $M = \frac{A(i,j)}{\sum_{i=1}^{n} A(i,j)}$, $M := M_{\exp} = Expand(M,e) = M^e$

(3) Inflate: $M := M_{\inf} = Inflate(M,r) = \frac{M(i,j)^r}{\sum_{i=1}^{n} M(i,j)^r}$

(4) Output the clusters by alternating expansion and inflation until convergence.

Selection Project Strategy

The selection algorithm (Tompa *et al*., 2005) iteratively removes k ($0 \leq k \leq d$) non-repetitive bases randomly from the l-mers with the bases left forming a set of ($l - k$)-mers. The selection remove in each time is accompanied by the generation of a selection function f, which would be applied to all the l-mers. The possible situation is shown as follows. The left ($l - k$)-mers would be the same after multiple l-mers going through selection remove. We put the l-mers with the identical bases in ($l - k$) positions into the same bucket. As long as it is possible to ensure that the l-mers in the same bucket have a great probability of being the motif instances, a qualified bucket can be produced.

Selection Parameters Setting

Setting proper selection parameters would ensure that the motif instances would be put into the same qualified bucket with a higher probability. The details of setting selection parameters are shown as follows:

Selection Size k

The value k affects the efficiency of the algorithm. The k value is greater, the shorter is the ($l - k$)-mer. In this condition, a lot of l-mers are selected into a bucket, discharging huge noise, which hinders the location of motif. On the contrary, the smaller the k value, the longer the ($l - k$)-mer. In this condition, as a small portion of l-mers is selected into a bucket, the qualified buckets with limited number would impede local further refinement. Theoretically, the maximum number of buckets is 4^{l-k} and the number of l-mers is $t \times (n - l + 1)$, so the amount of l-mers in a bucket on average from background sequence is:

$$E(l,k) = \frac{t \times (n-l+1)}{4^{l-k}}$$

where, t is the sequence number and n is the sequence length. For our expectation, the fewer the buckets from the amount of l-mers in background sequences, the better, where $E(l, k) < 1$.

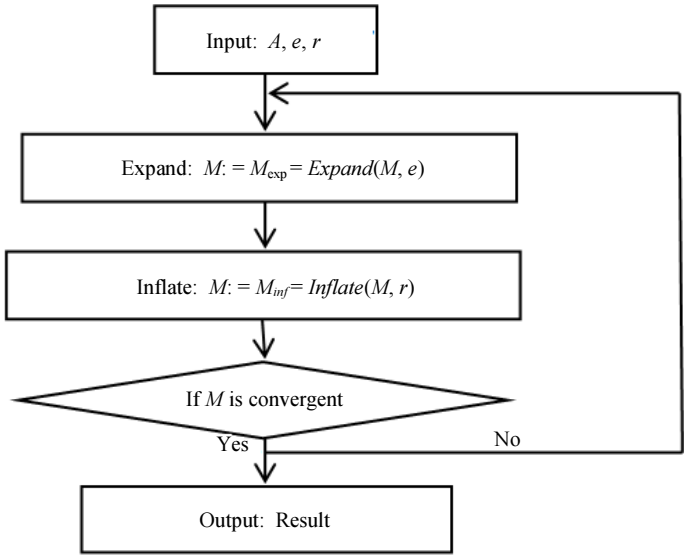


Fig. 2: The flowchart of MCL clustering algorithm

If $E(l, k) \leq 0.9$, then:

$$k \leq l - \log_4\left[\frac{t \times (n-l+1)}{0.9}\right]$$

and in this paper, we set that:

$$k = l - \log_4\left[\frac{t \times (n-l+1)}{0.9}\right]$$

Selection Number Num

When we select the value *num*, we shall ensure that sufficient number of motif instances can be stored in the bucket where motif is contained. The probability of a motif instance being selected to a qualified bucket is:

$$p_k = \frac{C_{l-d}^{l-k}}{C_l^{l-k}}$$

If selection is performed once, the probability of no more than a motif instances from t motif instances being put into a qualified bucket is:

$$p(x \leq a) = \sum_{i=0}^{a} C_t^i p_k^t (1-p_k)^{t-i}$$

So when the selection is performed *num* times, the probability of a motif instance chosen to be put into a qualified bucket at least once is:

$$p_{num} = p(count \geq 1) = 1 - p(count = 0) = 1 - [p(x \leq a)]^{num}$$

In order to ensure that there are enough motif instances in qualified buckets, it can be set that $p_{num} = p(count \geq 1) \geq 0.95$ and then:

$$num = \frac{\log(1-p_{num})}{\log(p(x \leq a)}$$

SMCLR Algorithm

Motif discovery is to search a number of similar base fragments in a set of DNA data and to maximize the information content of the position frequency matrix composed by these base fragments. Based on this idea, we propose a selection-based MCL clustering refinement algorithm (SMCLR), which mainly refers to establishing qualified buckets and refining qualified buckets.

By selecting reference sequence and using selection algorithm to construct a number of qualified buckets, we can reduce the solution space and only a limited number of conservative buckets are reserved as the initial state for further MCL clustering refinement. In order to make each qualified bucket converges to an optimal solution quickly and improve the predict accuracy, we apply MCL clustering algorithm to refine the qualified buckets. And then output the motif model with the maximum value of *IC*. The specific details of each section are as follows:

Step1: Construct Qualified Bucket

If we combine all the *l*-mers of the DNA dataset to construct buckets as the initial state, it will be time-consuming with the generation of a large number of redundant background *l*-mers in the bucket, as there are $(n - l + 1)^t$ possible alignments of all *l*-mer. Therefore, it is particularly important about how to select a good initial state for MCL refinement with a quick convergence to an optimal solution. In this paper, we construct a number of qualified buckets by selecting a reference sequence and adopting a selection project algorithm. The detailed procedure is as follows.

We select a reference sequence to divide the whole dataset into several independent subsets and expect to find the optimal solution of each subset. As there is no idea of which *l*-mer in a sequence being the motif instance, we choose a sequence as a reference one. In this reference sequence, all the *l*-mers are regarded as reference subsequences to look for motif instances that probably exist in other sequences. Generally, we select the first sequence S_1 as the reference sequence, then all the *l*-mer $x_{1j} = S_{1,j}\, S_{1,j+1} \ldots S_{1,\, j+l-1}$, $j = 1, 2, \ldots, n - l + 1$ in S_1 are regarded as reference subsequences. As the Hamming distance between two motif instances of the same motif must be less than or equal to $2d$; that is to say, given a motif x and two motif instances x_1, x_2, the result that $d_H(x_1, x_2) \leq 2d$ would be true. Let $B(x_{1j}, S_i)$ $i = 2, 3, \ldots, t$ represent the set of *l*-mers y in the i th sequence S_i such that $d_H(x_{1j}, y) \leq 2d$, that is:

$$B(x_{1j}, S_i) = \left\{y : y \in S_i, d_H(x_{1j}, y) \leq 2d\right\}$$

A bucket $C(x_{1j}, S)$, $j = 1, 2, \ldots, n - l + 1$ represents all the set of *l*-mer y in the whole sequences such that d $d_H(x_{1j}, y) \leq 2d$, that is:

$$C(x_{1j}, S) = \bigcup_{i=2}^{t} B(x_{1j}, S_j) \cup \{x_{1j}\}$$

As is known for all, the reference subsequences x_{1j} in the reference sequence, the true motif instances certainly exist in one of these $n - l + 1$ buckets $C(x_{1j}, S)$, but it is time-consuming to refine all the $n - l + 1$ buckets because most of these buckets have redundant background *l*-mers. Therefore, we set the threshold *max-size* and *min-size* to filter the interference buckets and retain the buckets which contain sufficient motif instances but less background *l*-mers. Under the OOPS model, each sequence has exactly one motif instance, the total number of motif instances equals the number of sequences t. Let *max-size* = $4t/3$, *min-size* = $2t/3$, for

each bucket $C(x_{1j}, S)$ according to the number of *l*-mers in each bucket $C(x_{1j}, S)$, denoted as $|C(x_{1j}, S)|$, it corresponds to three cases: (1) if $|C(x_{1j}, S)| < min\text{-}size$ holds, $C(x_{1j}, S)$ could be discarded; (2) if $min\text{-}size < |C(x1j, S)| <$ max-size holds, $C(x_{1j}, S)$ is considered as a qualified bucket $C_{valid}(x_{1j}, S)$; (3) if $|C(x_{1j}, S)| > max\text{-}size$ holds, then we use selection project strategy to construct qualified bucket $C_{valid}(x_{1j}, S)$ from $C(x_{1j}, S)$, where the number of *l*-mers satisfies min-size <| $C_{valid}(x_{1j}, S)$| < max-size.

Step2: Refine Qualified Bucket

We use MCL clustering algorithm and combine the sliding window mechanism as well as the dimensionality reduction strategy to refine the qualified buckets $C_{valid}(x_{1j}, S)$. In the following, the details involved in MCL clustering algorithm are introduced.

Similarity Measure

We map each *l*-mer in the qualified bucket $C_{valid}(x_{1j}, S)$ to a vertex on the graph G. In order to make the vertices corresponding to motif instances become more dense in the graph (as the MCL clustering always gathers the vertices with high similarity), we design the similarity of *l*-mer $x_1$and x_2 as follows:

$$Sim(x_1, x_2) = \begin{cases} \dfrac{len(x_1, x_2)}{2l - len(x_1, x_2)} & if \;\; d_H(x_1, x_2) \le 2d \\ 0 & if \;\; d_H(x_1, x_2) > 2d \end{cases}$$

where, $len(x_1, x_2) = l - d_H(x_1, x_2)$.

We set a threshold T_{sim}. If the similarity of two *l*-mer x_1 and x_2 is not more than T_{sim}, that is $T_{sim} \le Sim(x_1, x_2)$, the corresponding vertices of the two *l*-mers are connected by an edge and the weight of the edge is $Sim(x_1, x_2)$. Otherwise, the weight of the edge is zero. On this basis, an adjacent matrix *A* of the graph G is created. In this paper, we set $T_{sim} = 0.5$.

Sparse and Reduce Dimension

In order to overcome the overlapping of the motifs produced by MCL clustering, we introduce the sliding window scanning mechanism, with the size of the sliding window being set as *l*. Each row of adjacent matrix is sequentially scanned and for each l adjacent elements in every row, the number of the element not being zero is not greater than l, but if the number is greater than 1, then the element with the greatest weight among the l elements would be reserved. After sliding window scanning, the adjacent matrix is sparse. We further reduce the rank of the adjacent matrix, making the matrix of tens of thousands of ranks shrink to thousands of or even less. This process effectively compresses the solution space and improves the efficiency of clustering. First, we set a certain threshold for all the values of vertexes, so if the value of vertex is greater than or equal to this threshold, this vertex is reserved, or the vertex and all the associated edges will be directly removed. According to the rank of the initial matrix, the threshold is generally set to 5-20. The matrix with sparsity and reduced-dimension conducted is used as the initial matrix of the MCL clustering. After normalization, the iteration of Expand and Inflation shall be carried out until the matrix converges. After the MCL clustering refinement, the qualified bucket with the maximum information entropy is regarded as motif.

Based on the two steps, the whole SMCLR algorithm is described as follows:

SMCLR algorithm
Input: *l*, *d*, $S = \{S_1, S_2, ..., S_t\}$
Output: (*l*, *d*) motif X_{motif}
1: **for** each *l*-mer $x_{1,j}$ in S_1 **do**
2: $C(x_{1,j}, S) \leftarrow \Phi$
3: **for** $i \leftarrow 2$ to *t* **do**
4: $B(x_{1,j}, S_i) \leftarrow \Phi$
5: for each *l*-mer $x_{i,k}$ in S_i **do**
6: **if** $d_H(x_{1,j}, x_{i,k}) \le 2d$, **then** $B(x_{1,j}, S_i) \leftarrow B(x_{1,j}, S_i) \cup \{x_i, k\}$
7: $C(x_{1,j}, S) \leftarrow C(x_{1,j}, S) \cup B(x_{1,j}, S_i)$
8: Set thresholds to generate qualified bucket $C_{valid}(x_{1,j}, S)$
9: Use MCL clustering to $C_{valid}(x_{1,j}, S)$, compare IC
10: get X_{motif} from IC_{max}

Lines 2 – 7 describe the process of constructing qualified buckets. Line 8 describes the filtration of the qualified buckets. Lines 9-10 describe the MCL clustering refinement of the qualified buckets and the verification of the motif with the maximum IC score.

Results and Discussion

We use simulated data and the real data to test the performance of SMCLR algorithm. The simulated data is used to verify effectiveness and efficiency of our algorithm and the real data to verify its validity. According to (Buhler and Tompa, 2002), we generate the simulated data as follows: Firstly, we randomly produce 20 independent and identically distributed DNA sequences with a length of 600 and a motif *M* with a length of *l*; Secondly, we randomly select d different bases in *M* and replace the selected bases with other bases selected randomly and then generate 20 motif instances; Third, we plant the 20 motif instances in turn in the position randomly selected in the 20 DNA sequences.

In this study, the nucleotide level performance coefficient (*nPC*) (Crooks *et al*., 2004) is used to evaluate the motif prediction accuracy:

$$nPC = \frac{nTP}{nTP + nFN + nFP}$$

where, *nTP* is the number of nucleotide positions in both published motif sites and predicted motif sites; *nFN* is the number of nucleotide positions in published motif sites but not in predicted motif sites; *nFP* is the number of nucleotide positions not in published motif sites but in predicted motif sites. The range of value for *nPC* is from 0 to 1, the greater the value, the higher the predicted accuracy.

Results on Simulated Data

For each specific (*l*, *d*) problem instance, the predicted accuracy comes from the average result of 10 trials of simulated data experiments. We compare the performance of SMCLR algorithm with that of other widely used motif discovery algorithms, such as Projection, MEME, MCL-WMR and VINE.

As shown in Table 1, for the problem instances like (11, 2), (12, 3) and (15, 4), the predict accuracy of SMCLR algorithm can achieve 98%, 97% and 96% respectively. The performance is close to optimal. For the challenging problem instances like (13, 4), (17, 5) and (18, 6), the predict accuracy achieves 91%, 92% and 94%, which are superior to the results of Projection: 65%, 91% and 80%; the results of MEME: 62%, 82% and 79%; the results of MCL-WMR: 71%, 80% and 85%. VINE is nearly the best graph approximate algorithm for PMS at present, while the predicted accuracy of SMCLR algorithm is slightly higher than that of VINE. In terms of running time, MEME takes only a few seconds, while Projection requires much more, which takes over 20 min in problem instances (13, 4), (17, 5) and (18, 6); and MCL-WMR is stable in time running in different (*l*, *d*) instances. VINE could finish its execution in 10 min. For different (*l*, *d*) instances, SMCLR algorithm could control its operation in a reasonable time. Taking problem instance (18, 6) for example, SMCLR algorithm only uses 4.8 minutes and only 1.8 minutes for problem instance (15, 4), so the SMCLR algorithm strikes a good balanced between the running time and the predicted accuracy for solving PMS.

Results on Real Data

Firstly, we test the validity of SMCLR algorithm on the following five real data: Preproinsulin, Dihydrofolate reductase (DHFR), c-fos, metallothionein and the Yeast ECB. These are widely used in the current motif discovery algorithms to identify the real motifs. The five datasets have the following features: long in sequence, few in the number of sequence and with each sequence coming from different biological species as well as at least one motif instance found in each sequence. In testing, *l* and d ($d < l/2$) are constantly changed to check whether the proposed algorithm can find the known motifs using the specific (*l*, *d*). The predicted motifs and their corresponding predicted accuracy of these five real data sets are shown in Table 2, which indicates that SMCLR algorithm can work well on these five datasets. The underlined part of each predicted motif represents the part overlapped with the published motif. Note that, many existing recognition algorithms (Pavesi *et al*., 2001; Boucher, 2007; Lawrence *et al*., 1993; Buhler and Tompa, 2002) also test their validity on these five data sets. Since all of these algorithms (including SMCLR) show a good performance on these five data sets, here we do not make comparisons. In addition, Table 3 shows the sequence logos (Crooks *et al*., 2004) of the predicted motif, which graphically shows the degree of motif conservation measured by relative entropy.

Table 1: The results of different algorithms on different (*l*, *d*) problem instance

	nPC				
(*l*, *d*)	Projection	MEME	VINE	MCL-WMR	SMCLR
(11, 2)	0.91(10 s)	0.67 (5s)	0.95 (10s)	0.70 (12s)	0.98 (8s)
(12, 3)	0.74 (6m)	0.84 (5s)	0.92 (7.1m)	0.83(6.8m)	0.97(2.3m)
(13, 4)	0.65(43m)	0.62 (6s)	0.89 (6.8m)	0.71(9.7m)	0.91 (5.6m)
(15, 4)	0.93 (8m)	0.89 (6s)	0.96(6.5m)	0.91(5.3m)	0.96 (1.8m)
(17, 5)	0.91(21m)	0.82(6s)	0.90(4.3m)	0.80 (7.8m)	0.92 (3.6m)
(18, 6)	0.80 (33m)	0.79 (6s)	0.93(6.7m)	0.85 (18m)	0.94 (4.8m)

Table 2: Experiment result of SMCLR on real data

Data set	Predicted motif	Published motif	*nPC*	(*l*, *d*)
c-fos	CCAAATTAG	CCATATTAG	0.78	(9, 2)
Preproinsulin	GCCTCAGCCCCTT	GCCTCAGCCCCCA	0.73	(13, 2)
Yeast ECB	TATTTCCCAATAAGGAA	TTTCCCNNTNAGGAAA	0.65	(16, 3)
DHFR	ATTTCGTGGCA	TTTCGCGCCA	0.82	(11, 2)
metallothionein	CTCTGCGCGCCGCCC	CTCTGCGCRCCGCCC	0.86	(15, 2)

Table 3: Sequence logos of the predicted motifs

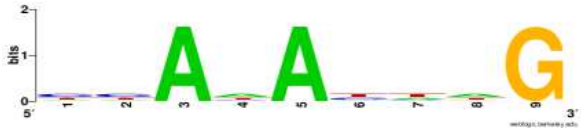
Data set	Predicted motif
Preproinsulin	
Metallonthionein	
c-fos	
DHFR	
Yeast ECB	

Table 4: The results for the five real data sets

	nPC			
Data set	MEME	VINE	MCL-WMR	SMCLR
CREB	0.89(1.72S)	0.42(44.20s)	0.78(16.08s)	0.82(18.04s)
MEF2	0.84(1.98s)	0.63(36.70s)	0.74(15.72s)	0.78(76.42s)
MYOD	0.68(2.30s)	0.57(30.08s)	0.60(12.24s)	0.79(32.27s)
SRF	0.86(2.14s)	0.74(46.82s)	0.83(27.81s)	0.92(96.52s)
TBP	0.78(40.08s)	0.68(78.24s)	0.70(84.49s)	0.76(11.05s)

Secondly, we use another five real DNA data to test the prediction performance of SMCLR algorithm: CREB, MEF2, MYOD, SRF and TBP, which choose from ABS database and consist of sequences from different species of labeled regulatory binding sites. As is shown in Table 4, we give the performance coefficient (nPC) and the running time of MEME, VINE, MCL-WMR and SMCLR. Through the test of these five sets of real data, it can be found that the prediction accuracy of SMCLR is better than that of MEME, VINE and MCL-WMR on some data sets (i.e., MYOD and SRF), but worse than on the other data sets (i.e., CREB, MEF2 and TBP). That is to say the SMCLR algorithm can not only effectively solve the motif discovery problem of different length motifs in different length sequences, but also show good performance compared with the existing widely used motif discovery algorithm and improve the operation efficiently. And the phenomenon provides guidance for identifying motifs. The predicted motif of different algorithms can complement each other to help to improve the predict accuracy.

Conclusion

In this study, we propose a MCL clustering refinement algorithm based on selection project to discover TFBSs with high prediction accuracy in a relatively short time. Firstly, we divide the dataset into several subsets by selecting reference sequence and using selection algorithm. Then we use a powerful data clustering method MCL cluster to identify motifs. Experiment results on simulated data show that the proposed algorithm can effectively solve planted (*l*, *d*) motif problems in a reasonable time, superior to the compared algorithms in prediction accuracy. Moreover, for the experiments on real biological data, we use two groups of data sets: (1) The first group includes Preproinsulin, Dihydrofolate reductase (DHFR), c-fos, metallothionein and the Yeast ECB, which are used by many existing recognition algorithm to test their validity. For each of these data sets, SMCLR is able to find all or a large part of TFBSs. (2) the second group includes CREB, MEF2, MYOD, SRF and TBP, which consist of

sequences from different species of labeled regulatory binding sites. The validity of the proposed algorithm is tested on these fivereal data sets.

Acknowledgements

This work was supported by the key research and development project of Shaanxi China (2018ZDCXL–NY–02–03).

Author's Contributions

Chunxiao Sun and Zhiyong Zhang: Designed and performed the experiments, analyzed the data and prepared the paper.

Jinglei Tang: Performed gene cloning experiments and constructed the recombinant plasmids.

Shuai Liu: Designed the experiments and revised the manuscript.

Ethics

The authors declare their responsibility for any ethical issues that may arise after the publication of this manuscript.

Conflict of Interest

The authors declare that they have no competing interests. The corresponding author affirms that all of the authors have read and approved the manuscript.

References

Bailey, T.L., N. Williams and C. Misleh, 2006. MEME: discovering and analyzing DNA and protein sequence motifs. Nucleic Acids Res., 34: W369-W373.

Bandyopadhyay, S., S. Sahni and S. Rajasekaran, 2012. PMS6: A fast algorithm for motif discovery. Proceedings of the IEEE International Conference on Computational Advances in Bio and Medical Sciences. Las Vegas, NV, pp: 1-6.

Boucher, C., 2007. A graph clustering approach to weak motif recognition. Proceedings of the 7th Workshop on Algorithms in Bioinformatics (WABI '07). Lecture Notes in Computer Science Springer, Berlin/Heidelberg, pp: 149-160.

Boucher, C., D. Brown and P. Church, 2007. A graph clustering approach to weak motif recognition. Proceedings of the 7th International Workshop on Algorithms in Bioinformatics. Philadelphia, PA, USA, pp: 149-160.

Buhler, J. and M. Tompa, 2002. Finding motifs using random projections. J. Comput. Biology, 9: 225-242.

Crooks, G.E., G. Hon and J.M. Chandonia, 2004. WebLogo: A sequence Logo generator. Genome Res., 14: 1188-1190.

Das, M. and H. Dai, 2007. A survey of DNA motif finding algorithms. BMC Bioinformatics, 8: S21.

Davila, J., S. Balla and S. Rajasekaran, 2007. Fast and practical algorithms for planted (*l*, *d*) motif search. IEEE/ACM Trans. Computational Biology Bioinformatics, 4: 544-552.

Dinh, H., S. Rajasekaran and J. Davila, 2012. qPMS7: A fast algorithm for finding (*l*, *d*)-motifs in DNA and protein sequences. PLoS One, 7: e41425.

Dinh, H., S. Rajasekaran and V.K. Kundeti, 2011. PMS5: An efficient exact Algorithm for the (*l*, *d*) motif finding problem. BMC Bioinformatics, 12: 410.

Eskin, E. and P.A. Pevzner, 2002. Finding composite regulatory patterns in DNA sequences. Bioinformatics, 18: 354-363.

Evans, P.A. and A. Smith, 2003. Toward optimal motif enumeration. Proceedings of the 8th International Workshop Algorithms and Data Structures, Ottawa, pp: 47-58.

Fratkin, E., B.T. Naughton and D.L. Brutlag, 2006.MotifCut: regulatory motifs finding with maximumdensity subgraphs. Bioinformatics, 22: e150-e157.

Hertz, G.Z. and G.W. Hartzell, 1990. Identification of consensus patterns in unaligned DNA sequences known to be functionally related. Computer Application Biosciences Cabios, 6: 81-92.

Ho, E.S., C.D. Jakubowski and S.I. Gunderson, 2009. iTriplet, a rule-based nucleic acid sequence motif finder, Algorithms Molecular Biology, 4: 14.

Jones, N. and P. Pevzner, 2004.An introduction to bioinformatics algorithms, Cambridge: MIT Press.

Lawrence, C.E., S.F. Altschul and M.S. Boguski, 1993. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment, Science, 262: 208-214.

Liu, X., D.L. Brutlag and J.S. Liu, 2001. BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. Pacific Symposium Biocomputing, 6: 127-138.

Marsan, L. and M.F. Sagot, 2000. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. J. Computational Biology, 7: 345-362.

Neuwald, A.F., J.S. Liu and C.E. Lawrence, 1995. Gibbs motif sampling: detection of bacterial outer membrane protein repeats. Protein Sci., 4: 1618-1632.

Pavesi, G., G. Mauri and G. Pesole, 2001. An algorithm for finding signals of unknown length in DNA sequences. Bioinformatics, 17: 207-214.

Pavesi, G., P. Mereghetti and G. Mauri, 2004. Weeder web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes, Nucleic Acids Res., 32: W199-W203.

Pisanti, N., A.M. Carvalho and L. Marsan, 2006. RISOTTO: fast extraction of motifs with mismatches. Proceedings of the 7th Latin American Symposium: Theoretical Informatics, Valdivia, pp: 757-768.

Rajasekaran, S., S. Balla and C. Huang, 2005. Exact algorithms for planted motif problems. J. Computat. Biology, 12: 1117-1128.

Stormo, G.D., T.D. Schneider and L. Gold, 1982. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in *E. coli*. Nucleic Acids Res., 10: 2997-3011.

Tanaka, S., 2014.Improved exact enumerative algorithms for the planted (*l*, *d*)-motif search problem. IEEE/ACM Trans. Computational Biology Bioinformatics, 11: 361-374.

Tompa, M., N. Li and T.L. Bailey, 2005.Assessing computational tools for the discovery of transcription factor binding sites. Nature Biotechnol., 23: 137-144.

van Dongen, S., 2000. Graph clustering by flow simulation. PhD thesis Centers for Mathematics and Computer Science (CWI), University of Utrecht.