An Efficient Algorithm for Discovering Motifs in Large DNA Data Sets

Qiang Yu, Hongwei Huo*, *Member, IEEE*, Xiaoyang Chen, Haitao Guo, Jeffrey Scott Vitter, *Fellow, IEEE*, and Jun Huan, *Member, IEEE*

***Abstract*—The planted (l, d) motif discovery has been successfully used to locate transcription factor binding sites in dozens of promoter sequences over the past decade. However, there has not been enough work done in identifying (l, d) motifs in the next-generation sequencing (ChIP-seq) data sets, which contain thousands of input sequences and thereby bring new challenge to make a good identification in reasonable time. To cater this need, we propose a new planted (l, d) motif discovery algorithm named MCES, which identifies motifs by mining and combining emerging substrings. Specially, to handle larger data sets, we design a MapReduce-based strategy to mine emerging substrings distributedly. Experimental results on the simulated data show that i) MCES is able to identify (l, d) motifs efficiently and effectively in thousands to millions of input sequences, and runs faster than the state-of-the-art (l, d) motif discovery algorithms, such as F-motif and TraverStringsR; ii) MCES is able to identify motifs without known lengths, and has a better identification accuracy than the competing algorithm CisFinder. Also, the validity of MCES is tested on real data sets. MCES is freely available at http://sites.google.com/site/feqond/mces.**

***Index Terms*—ChIP-seq, emerging substrings, MapReduce, motif discovery.**

I. Introduction

MOTIF discovery is an important and challenging problem in computational biology. It plays a key role in locating transcription factor binding sites (TFBS) in DNA sequences. Binding sites tend to be short and degenerate, so it is difficult to distinguish them from the input sequences. The planted (l, d) motif discovery [1] is a famous formulation for motif discovery, which has been proven to be NP-complete [2].

Planted (l, d) Motif Discovery Problem: Given a set of n-length DNA sequences $\{s_1, s_2, \ldots, s_t\}$ over the alphabet $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ and two nonnegative integers l and d, satisfying $0 \leq d < l < n$, the task is to find one or more l-length strings m such that m occurs in all or a large fraction of the sequences with up to d mismatches. The l-length string m is called a (l, d) motif and each occurrence of m is called a motif instance of m.

According to how and where motif occurrences appear in the sequences, there are three types of motif discovery sequence model: OOPS, ZOOPS and TCM [3], corresponding to one occurrence per sequence, zero or one occurrence per sequence and zero or more occurrences per sequence, respectively. The ZOOPS and TCM sequence model are more consistent with the real biological situation than the OOPS model, but identifying motifs under these two models is more difficult than that under the OOPS model.

Numerous algorithms have been proposed to identify motifs in several to dozens of promoter sequences from co-regulated or homologous genes [4]. These algorithms can be divided into two categories in terms of the used motif representation models: those using consensus sequences [5] and those using position weight matrices (PWM) [6]. Most identification algorithms based on consensus sequences are pattern-driven [7]–[11]. They traverse all sequence patterns of length l with an initial search space of $O(4^l)$ and report all (l, d) motifs. The identification algorithms based on PWM usually employ statistical techniques [3], [12]. They iteratively update an initial PWM and report the motif with high score.

In recent years, the novel experimental techniques, such as protein-binding microarray (PBM) [13] and chromatin immunoprecipitation followed by high-throughput sequencing (ChIP-seq) allows the genome-wide identification of TFBSs [4], [14]. The experiments can output a list of transcription factor binding regions (i.e., peak regions), but motif discovery methods are still needed to accurately locating TFBSs in these peak regions. The advantage of a ChIP-seq data set is that the sequences are cleaner than the traditional promoter sequences [4]. That is, not only a high percentage of sequences contain TFBSs, but also each sequence has a high resolution (i.e., the sequence length is short, about 200 base pairs). It seems easier for motif discovery methods to obtain a high identification accuracy in ChIP-seq data sets, but the size of a ChIP-seq data set is very large and the set contains thousands or more sequences, requiring a high computational efficiency of motif discovery. Unfortunately, almost all algorithms designed for identifying motifs in promoter sequences, either the pattern-driven algorithms or statistical algorithms, become too time-consuming for ChIP-seq data sets.

ChIP-tailored versions of traditional motif discovery algorithms have been proposed, such as MEME-ChIP [15]. These algorithms usually present limitations on the data set size by selecting a small subset of the sequences to make motif identification [16]. For example, MEME-ChIP just selects 600 sequences at random from the input sequences and then identifies motifs

Manuscript received March 27, 2015; accepted March 31, 2015. Date of publication April 09, 2015; date of current version August 05, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61173025 and 61373044, and the Fundamental Research Funds for the Central Universities under Grant JB150306 and XJS15014. *Asterisk indicates corresponding author.*

Q. Yu, X. Chen, and H. Guo are with the School of Computer Science and Technology, Xidian University, Xi'an, 710071, China (e-mail: qyu@mail.xidian.edu.cn; xychen@mail.xidian.edu.cn; htguo@mail.xidian.edu.cn).

*H. Huo is with the School of Computer Science and Technology, Xidian University, Xi'an, 710071, China (e-mail: hwhuo@mail.xidian.edu.cn).

J. S. Vitter and J. Huan are with the Information and Telecommunication of Technology Center, The University of Kansas, Lawrence, 66047, USA (e-mail: {jsv,jhuan}@ku.edu).

Digital Object Identifier 10.1109/TNB.2015.2421340

by using the expectation-maximization algorithm. In spite of this, these algorithms still show a poor time performance due to maintaining the original algorithm framework. In contrast to MEME-ChIP, EXTREME [17] achieves a much better time performance by using the online expectation-maximization algorithm, but it requires too much storage space in handling large input (e.g., it requires about 8 GB memory for 10 Mb inputs). A few new algorithms [18], [19] are designed either based on suffix tree or De Bruijn graph, but they show poor time performance with the increase of l and d. Although DREME [20] can analyze very large data sets in minutes, it can only find short motifs. To process full-size ChIP-seq data sets efficiently, some algorithms based on word counting are proposed, such as RSAT [21] and CisFinder [22]. Both RSAT and CisFinder just take advantages of frequencies of very short words for the sake of good time performance, so they may miss some useful information contained in the sequences; also, CisFinder does not support outputting motifs of a specified length.

Against the background of identifying motifs in ChIP-seq data sets, it is necessary to design new algorithms with the following features: i) they can handle the full-size input sequences and make full use of the information contained in the sequences, ii) they can complete the computation with a good time performance and a good identification accuracy, iii) they can identify motifs without the OOPS constraint and iv) they can report motifs with or without a specified length.

To cater these needs, we proposed a new motif discovery algorithm, named MCES, based on mining and combining emerging substrings, which are potential (l, d) motif instances. We design MCES in terms of the ZOOPS sequence model. To handle very large data sets, we also design a MapReduce-based strategy to mine emerging substrings distributedly. MCES fully uses the emerging substrings of different lengths, and is able to efficiently and effectively identify motifs with or without a specified length in full-size ChIP-seq data sets.

The rest of the paper is organized as follows. Section II first gives the overview of the proposed algorithm, then describes the mining step and the combining step in detail, and finally shows the whole algorithm. Section III presents the results and discussion. We conclude the paper in Section IV.

II. Methods

A. Overview

We first introduce an observation that a given instance m' of a (l, d) motif m may exactly occur multiple times in ChIP-seq data sets. ChIP-seq data sets contain thousands or more DNA sequences, and thus the (l, d) motif m also has thousands or more instances. Since each instance differs from m in at most d positions, we can expect to find some motif instances repeating multiple times in thousands of sequences. In Section III-A we confirm this observation by using probabilistic analysis and demonstrate that motif instances have higher occurrence frequencies than the background l-mers.

In view of these considerations, we identify motifs by mining and combining substrings with high occurrence frequencies. Accordingly, our algorithm contains two main steps, namely the mining step and the combining step. Table I summarizes the notations used in this paper.

In the mining step, we mine substrings of different lengths ($l_{\min} \leq l \leq l_{\max}$) simultaneously, for i) the length of the identified motif is unknown in advance and ii) some segments of a motif are also over-represented and mining them helps us obtain more motif information. Moreover, to reduce the disturbance of random over-represented substrings, we perform mining by using both a test set and a control set of DNA sequences. The test set contains the sequences that share the motifs to be identified, whereas the control set only consists of the background sequences (i.e., the sequences that do not contain motif instances). Thereby, the interest substrings or motif instances are only over-represented in the test set rather than in the control set, and we call such substrings *emerging substrings*. Naturally, we convert our mining task to emerging substrings mining problem [23] as follows. The detailed description of the mining step is given in Section II-B.

TABLE I
Notations Used in This Paper.

Notation	Explanation
$\|x\|$	The length of a string, the number of DNA sequences in a data set or the size of a set.
l-mer	An l-length string over $\sum$ = {A, C, G, T}.
D, D_t and D_c	A data set of DNA sequences. $D = \{s_1, s_2, \ldots, s_{\|D\|}\}$ where each s_i is a DNA sequence. D_t and D_c denote a test set and a control set of DNA sequences, respectively.
$\|\|D\|\|$	The total length of sequences in D.
$count(\varphi, D)$	The occurrence count of a substring φ in D. $count(\varphi, D) = \|\{s_i: s_i \in D$ and φ is a substring of $s_i\}\|$. Note that, we calculate $count(\varphi, D)$ in terms of the ZOOPS sequence model, and thus we counts the number of sequences in D containing φ.
$freq(\varphi, D)$	The occurrence frequence of a substring φ in D. $freq(\varphi, D) = count(\varphi, D) / \|D\|$.
$growth(\varphi, D_t, D_c)$	The growth rate of a substring φ from D_c to D_t. $growth(\varphi, D_t, D_c) = freq(\varphi, D_t) / freq(\varphi, D_c)$, if $freq(\varphi, D_c) \neq 0$; $growth(\varphi, D_t, D_c) = \infty$ otherwise. A large value of the growth rate means that φ is highly discriminative for the two data sets.
$d_H(x, x')$	The Hamming distance between two l-mers x and x'.
$sim_{str}(\varphi_1, \varphi_2)$	The similarity of two strings φ_1 and φ_2 calculated by (4).
$sim_{pwm}(w_1, w_2)$	The similarity of two PWMs w_1 and w_2 calculated by (6).

Emerging Substrings Mining Problem: Given a test set D_t and a control set D_c of sequences over $\sum = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$, a threshold frequency $\rho_f (1/|D_t| \leq \rho_f \leq 1)$, and a minimum growth rate $\rho_g > 1$, the task is to find all substrings φ over $\sum$ such that $l_{\min} \leq |\varphi| \leq l_{\max}$, $freq(\varphi, D_t) \geq \rho_f$ and $growth(\varphi, D_t, D_c) \geq \rho_g$. The substrings satisfying the conditions of both ρ_f and ρ_g are called emerging substrings.

In the combining step, we combine emerging substrings together by using clustering methods to obtain predicted motifs. The key factors to consider are as follows. First, the mining step may output many emerging substrings due to mining substrings of different lengths, and these emerging substrings should be combined efficiently. Second, we need a method to calculate the similarity between two substrings of different lengths. Section II-C describes the combining step in detail.

B. Mining Step

We take advantage of the string mining algorithm proposed by Fischer, Heun and Kramer [23] to calculate the mining step efficiently. Their mining algorithm runs in optimal time. It first constructs the suffix array (SA) and the longest common prefix

array (LCP) for the input data sets, and then visits all substrings by simulating the suffix tree traversal in the SA using the LCP information.

To adjust their mining algorithm to closely fit our problem, we make the following improvements. First, since the occurrence frequencies of (l,d) motif instances are different for distinct length l, we set an adaptive threshold frequency ρ_f for each possible length l through probabilistic analysis, rather than using a fixed one. Second, we design a distributed version of the mining algorithm based on MapReduce to make it scale well for very large data sets.

1) Mining Parameters: The threshold frequency ρ_f and the minimum growth rate ρ_g are two important parameters in the mining step. We set ρ_g as 2 to reduce the disturbance of random over-represented substrings. In the following discussion, we focus on how to set the threshold frequency ρ_f.

In order to properly choose the threshold frequency ρ_f for mining potential (l,d) instances, we estimate the probability that a random instance m' of a (l,d) motif m occurs (or is implanted) in a given sequence, denoted by P_{occ}, based on the assumption that the given sequence contains one motif instance. Since it is easy to calculate the probability of the occurrence of m' given $d_H(m,m') = i (0 \le i \le d)$, denoted by $P(occ(m')|d_H(m,m') = i)$, P_{occ} can be calculated by using the theorem of total probability:

$$\begin{aligned} P_{occ} &= \sum_{i=0}^{d} P(d_H(m,m')=i) \times P(occ(m')|d_H(m,m')=i) \\ &= \sum_{i=0}^{d} P(d_H(m,m')=i) \times \frac{1}{\binom{l}{i} \times 3^i}. \end{aligned} \tag{1}$$

Furthermore, we need to estimate the probability of $d_H(m,m') = i (0 \le i \le d)$. Since $d_H(m,m') \le d$, there exist d positions in the alignment of m and m' covering the positions where m' differs from m. For each of the d positions, let p represents the probability that m' differs from m. Then, we calculate $P(d_H(m,m')=i)$ by using the binomial formula:

$$P(d_H(m,m')=i) = \binom{d}{i} p^i (1-p)^{d-i}. \tag{2}$$

In (2), the parameter p reflects the conservation of the (l,d) motif. For a given (l,d) motif m, a small p indicates that m is more conserved, namely m differs from its instances in fewer positions. We set p as 0.2, 0.5 and 0.8 to represent high conservation, intermediate conservation and low conservation, respectively.

On the above basis, we calculate the threshold frequency ρ_f by (3), where the parameter $q (0 \le q \le 1)$ represents the percentage of input sequences containing motif instances, for supporting the ZOOPS sequence model.

$$\rho_f = q P_{occ}. \tag{3}$$

In practice, to determine the value of the threshold frequency ρ_f for each length l, we make the following settings. We choose d corresponding to the challenging problem instances, and set both p and q as 0.8. When l is large $(l > 15)$, the value of ρ_f is too small (< 0.0001) and there are too many disturbed substrings passing the constraint; in this case, we reset ρ_f to a low bound 0.002, which is approximately equal to the value of ρ_f for $l = 15$.

2) MapReduce Strategy: In mining emerging substrings, although the storage space required by the associated data structures (i.e., SA and LCP) is linear in the size of input data sets, it will exceed the memory capacity of a single machine when the input data sets are very large. To solve this problem, we propose an emerging substrings mining method based on MapReduce [24], which facilitates us to develop scalable data-intensive application on distributed systems.

We partition the input data sets D_t and D_c to multiple data blocks D_i with the size z. More specifically, D_t is partitioned to $n1 = \lceil \|D_t\|/z \rceil$ blocks: $D_1, \ldots, D_{n1}$, and D_c is partitioned to $n2 = \lceil \|D_c\|/z \rceil$ blocks: $D_{n1+1}, \ldots, D_{n1+n2}$. Note that, the data blocks D_{n1} and D_{n1+n2} may have a size smaller than z.

In Map phase, each Map task deals with one data block D_i and returns the occurrence counts of all substrings φ in D_i using a set of 3-tuples $< \varphi, \alpha, \beta >$, where $< \varphi, \alpha, \beta >$ is equal to $< \varphi, count(\varphi, D_i), 0 >$ if D_i comes from D_t, $< \varphi, 0, count(\varphi, D_i) >$ otherwise. Here, using 3-tuples to represent the occurrence counts of substrings is for the convenience of merging them in Reduce phase.

Note that, the size of the intermediate results (i.e., the set of 3-tuples) generated from each map task could be very large; in that case I/O operations would consume too much computational time. To tackle this problem, we introduce a parameter $\lambda (0 < \lambda < 1)$ to filter out the substrings that do not satisfy the condition of the threshold frequency $\lambda\rho_f$. The parameter λ allows us to use relaxed constraint to filter out some unnecessary substrings in advance with interesting substrings retained. The effect of the parameter λ is demonstrated in Section III-C3. On the above basis, the map function is as follows, where traversing all substrings (line 2) is performed by the algorithm in [23].

Algorithm 1 Map function

Input: a data block D_i
Output: the set of 3-tuples T_i
1: $T_i \leftarrow \Phi$
2: **for** each substring φ **do**
3: get its occurrence count $count(\varphi, D_i)$
4: $freq(\varphi, D_i) \leftarrow count(\varphi, D_i)/|D_i|$
5: **if** $l_{\min} \le |\varphi| \le l_{\max}$ and $freq(\varphi, D_i) > \lambda\rho_f$ **then**
6: **if** D_i is from D_t **then**
7: add $< \varphi, count(\varphi, D_i), 0 >$ to T_i
8: **else**
9: add $< \varphi, 0, count(\varphi, D_i) >$ to T_i
10: **return** T_i

In Reduce phase, reduce task merges the 3-tuples with the same key (substring) into one 3-tuple by summing up the values in the corresponding positions. That is, when all map tasks finish the transmission of their intermediate results, for a substring φ its 3-tuples are $< \varphi, \alpha_1, \beta_1 >, < \varphi, \alpha_2, \beta_2 >, \ldots$, and then the final 3-tuple of φ is $< \varphi, \alpha_1 + \alpha_2 + \cdots, \beta_1 + \beta_2 + \cdots >$, which equals to $< \varphi, count(\varphi, D_t), count(\varphi, D_c) >$. Finally, reduce task outputs the substrings that satisfy the conditions of both ρ_f and ρ_g. The reduce function is as follows.

Algorithm 2 Reduce function

Input: all sets of 3-tuples $T_i(1 \leq i \leq n_{block})//n_{block}$ is the total number of D_i
Output: the set of emerging substrings S_{es}
1: $S_{es} \leftarrow \Phi$
2: **for** $i \leftarrow 2$ to n_{block} **do**
3: merge T_i to T_1
4: **for** each 3-tuple $< \varphi, \alpha, \beta >$ in T_1 **do**
5: $freq(\varphi, D_t) \leftarrow \alpha/|D_t|$
6: $freq(\varphi, D_c) \leftarrow \beta/|D_c|$
7: $growth(\varphi, D_t, D_c) \leftarrow freq(\varphi, D_t)/freq(\varphi, D_c)$
8: **if** $freq(\varphi, D_t) > \rho_f$ and $growth(\varphi, D_t, D_c) > \rho_g$ **then**
9: add φ to S_{es}
10: **return** S_{es}

C. Combining Step

1) Combining Method: We describe the main framework of the combining step in Algorithm 3 and accordingly give an example in Fig. 1 using the Esrrb data set [25]. The combining step includes two stages: clustering emerging substrings and clustering PWMs that correspond to the clusters of emerging substrings.

Algorithm 3 Combine Emerging Substrings

Input: the set of emerging substrings S_{es}
Output: the set of motifs M
1: $S_{ds} \leftarrow \Phi$ // the set of dispersive substrings
2: $S_{pwm} \leftarrow \Phi$ // the set of PWMs
3: $M \leftarrow \Phi$ // the set of motifs
4: **for** $i \leftarrow l_{\min}$ to $l_{\max}$ **do**
5: cluster the emerging substrings in S_{es} of length i
6: **for** each obtained cluster of emerging substrings C_{es} **do**
7: **if** $|C_{es}| > 5$ **then**
8: align the substrings in C_{es} and get a PWM w
9: add w to S_{pwm}
10: **else**
11: add the substrings in C_{es} to S_{ds}
12: cluster the substrings in S_{ds} and add obtained PWMs to S_{pwm}
13: cluster the PWMs in S_{pwm}
14: **for** each obtained cluster of PWMs C_{pwm} **do**
15: align the PWMs in C_{pwm} and get a combined PWM w
16: fetch the segment of w with high information content to obtain a motif m
17: add m to M
18: **return** M

All clustering operations are completed by using the MCL algorithm [26], which is a graph clustering algorithm based on simulation of flow in graphs and has been widely used in bioinformatics. In the graphical representation of the substrings/PWMs, each substring/PWM is represented by a node, and the weight of the edge between two nodes is the similarity between two substrings/PWMs defined in Section II-C2. There are two parameters required in using the MCL algorithm, namely the power parameter e and the inflation parameter r, and we set them as 2 and 1.8 following the suggestions given in [27].

In Stage 1 (lines 4 to 12 in Algorithm 3), to ensure good time performance, we iteratively cluster the emerging substrings of the same length, rather than cluster all the emerging substrings at one time. If an obtained cluster contains ≤ 5 substrings, we call it an invalid cluster and call the substrings in it *dispersive substrings*. The dispersive substrings of the same length show weak or no motif information, but all the dispersive substrings of different lengths may contain strong motif information. Thus, clustering the dispersive substrings of different lengths can help us to find the motif information missed in clustering the substrings of the same length. In Fig. 1, the symbol # represents the number of corresponding substrings. In all of the 787 emerging substrings, there are 93 dispersive substrings; in all of the 40 valid clusters, there are 9 ones coming from clustering the dispersive substrings.

For each valid cluster, we need to align the substrings in it to get a PWM. First, we can align two substrings by aligning their maximum overlap and obtain a score by (4). Second, we align all substrings in the cluster by building a maximum weight spanning tree using the Prim algorithm.

In order to implement the alignment along with building the spanning tree, we select the substring that is the cluster center as the start node of the spanning tree, which is located in the thickest parts of the graph of substrings. From Fig. 1, we can see that the cluster center (the bold substring) corresponds to the segment of PWM with high information content. Then, we repeatedly select a new node (substring) v linked to an already selected node (substring) u and align v to u, until all substrings are selected and aligned .

In Stage 2 (lines 13 to 18 in Algorithm 3), we cluster the obtained PWMs, which contributes to eliminating redundant motif information and combining parts of motif information to form whole motifs. For each cluster of PWMs, we align the PWMs in it. The aligning process is similar to that of aligning emerging substrings, except for using a different similarity measure provided by (6). As shown in Fig. 1, after aligning the PWMs in each cluster of PWMs, the corresponding substrings are also aligned, and then we get a combined PWM for each cluster of PWMs.

Finally, we obtain the motifs by fetching the segments of combined PWMs with high information content (the bold parts of the combined PWMs). Given a combined PWM w, when identifying motifs with a specified length l, we traverse all segments of length l and fetch the segment of w with maximum information content as the motif. When identifying general motifs (without a specified length), we obtain the motif by deleting the columns with low information content from two sides of w. More specifically, we find a column i and a column j of w such that for any $k < i$ and $k > j$, the information content of column k is smaller than a threshold (0.1), and we fetch the segment corresponding to columns i to j as the motif. As shown in Fig. 1, we obtain three motifs by clustering 40 PWMs (clusters of emerging substrings).

2) Similarity Measures: In the combining step, our algorithm needs to cluster two types of data elements, namely emerging

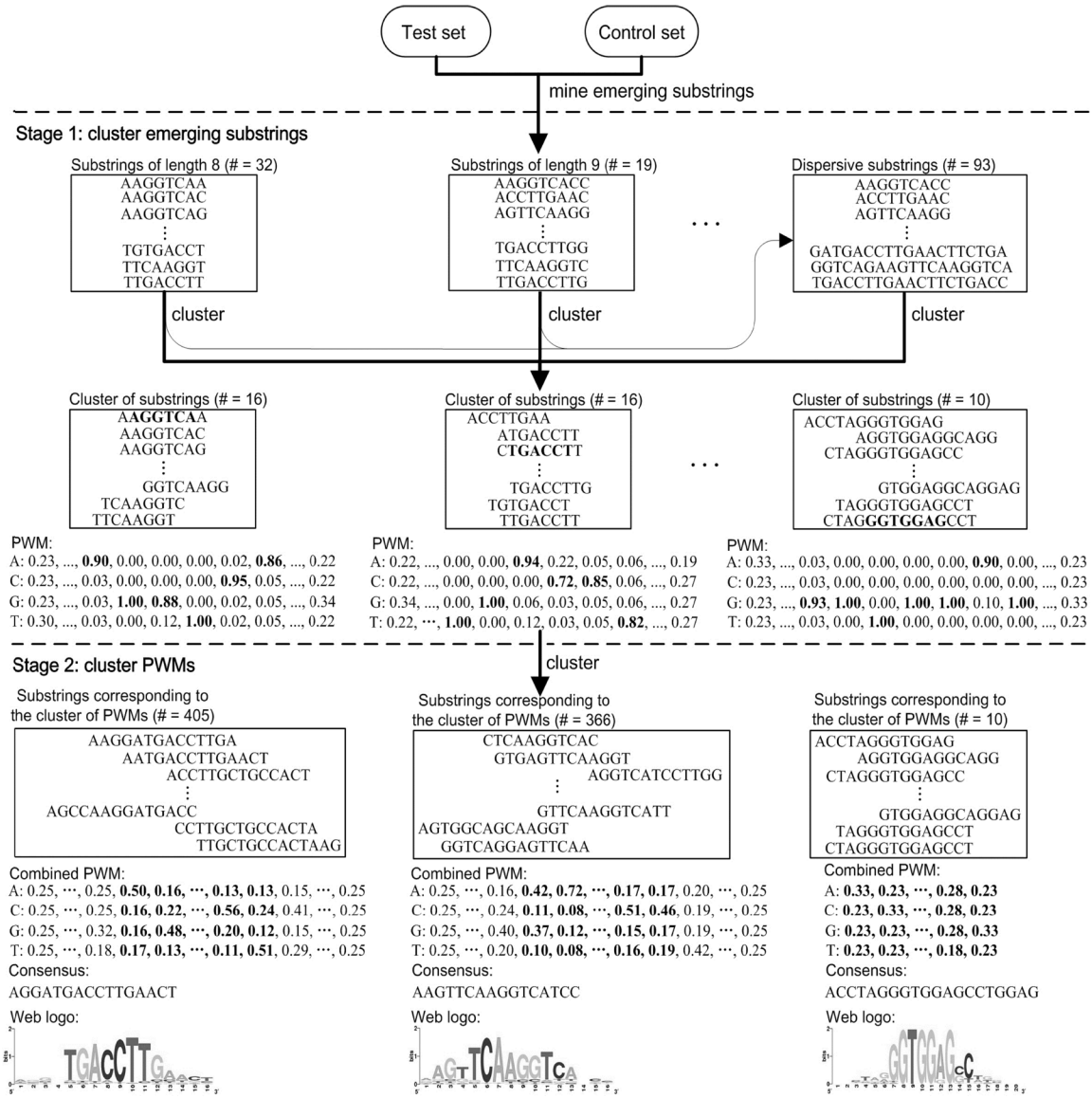

Fig. 1. Illustration of the combining step using the Esrrb data set.

substrings and PWMs. Therefore, we design two similarity measures for these two types of data elements.

First, we compare two given emerging substrings φ_1 and φ_2 in terms of their maximum overlap. The two emerging substrings may have different lengths and are not aligned. We expect the overlap of the two emerging substrings represents the common segment of motif occurrences. For the overlap of length l, we allow a maximum number of mismatches $l/4$; moreover, we say an overlap is valid only when its length is larger than four. Taking these considerations into account, we calculate the similarity of φ_1 and φ_2 by (4), where $len_{mo}(\varphi_1, \varphi_2)$ denotes the length of the maximum overlap of φ_1 and φ_2.

$$sim_{str}(\varphi_1, \varphi_2) = \begin{cases} \frac{len_{mo}(\varphi_1,\varphi_2)}{|\varphi_1|+|\varphi_2|-len_{mo}(\varphi_1,\varphi_2)} & \text{if } len_{mo}(\varphi_1, \varphi_2) > 4, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Second, we need to compare two given PWMs w_1 and w_2, which correspond to the sets of aligned similar emerging substrings. Comparing w_1 and w_2 needs to consider two aspects: i) how to compare two PWM columns $w_1[i]$ and $w_2[j]$, and ii) how to align w_1 and w_2 and sum the scores from comparing corresponding columns. We compare $w_1[i]$ and $w_2[j]$ by using the Average Log Likelihood Ratio (ALLR) to distinguish their probability distributions [28]:

$$ALLR(w_1[i], w_2[j]) = \frac{\sum_r n_{rj} \ln\left(\frac{f_{ri}}{b_r}\right) + \sum_r n_{ri} \ln\left(\frac{f_{rj}}{b_r}\right)}{\sum_r n_{ri} + n_{rj}} \quad (5)$$

where b_r is the background frequency of base $r \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$, and n_{ri}/n_{rj} and f_{ri}/f_{rj} are the count and frequency of base r at the ith/jth position of w_1/w_2.

For each PWM w obtained in Stage 1 of the combining step, there is a segment of w with significantly high information content, which is likely to correspond to a segment of the motif. To reduce the disturbance of columns with low information content, we compare w_1 and w_2 by finding aligned segments of w_1 and w_2 with maximum *ALLR*. Thus, we calculate the similarity

of w_1 and w_2 by (6), where l_s denotes the length of the segment (in practice, we set l_s as 6).

$$sim_{pwm}(w_1, w_2) = \max_{i,j} \left(\sum_{k=0}^{l_s-1} ALLR(w_1[i+k], w_2[j+k]) \right). \quad (6)$$

D. Whole Algorithm

The whole algorithm of MCES is described in Algorithm 4. MCES can make de nove motif discovery without any presets. Note that, mining appropriate amount of emerging substrings is sensitive to the threshold frequency ρ_f. In default, we set ρ_f for mining potential (l, d) instances as described in Section II-B1, and set the range of l as the usual motif lengths $8 \leq l \leq 25$. Optionally, users can reset ρ_f and the range of l to meet their own needs. We use a threshold z' (40 Mbytes in default) to determine whether to perform the mining step using MapReduce. After performing the mining step and the combining step (lines 2 to 6), if the set of motifs M is empty, we usually do not get enough emerging substrings in the mining step. In this case, we relax the threshold frequency ρ_f and reperform the mining step and the combining step (lines 7 to 9).

Algorithm 4 MCES

Input: a test set D_t and a control set D_c of DNA sequences
Output: the set of motifs M
1: set mining parameters
2: **if** $\|D_t\| + \|D_c\| \leq z'$ **then**
3: perform mining step in a single machine
4: **else**
5: perform mining step distributedly using MapReduce
6: perform combining step using Algorithm 1 and get M
7: **if** $M = \Phi$ **then**
8: $\rho_f \leftarrow \rho_f/2$
9: perform mining step and combining step again
10: **return** M

The time complexity of MCES depends on both the mining step and the combining step. In the mining step, all of the operations, namely constructing data structures (SA and LCP) and traversing all substrings, are completed in optimal time [23], namely $O(\|D_t\| + \|D_c\|)$. The combining step is mainly to apply the MCL clustering algorithm multiple times, and takes $O(LN^3)$ time: L denotes the number of times that the MCL algorithm runs, and N^3 is the time complexity of the MCL algorithm [25], where N denotes the number of nodes to be clustered; since both L and N are approximately equal to the sequence length n, we replace them with n in the time complexity. Therefore, the time complexity of MCES is $O(\|D_t\| + \|D_c\| + n^4)$. The space complexity of MCES mainly depends on the mining step, in which the data structures (not compressed SA and LCP) need a storage space of $O((\|D_t\|+\|D_c\|)\log(\|D_t\|+\|D_c\|))$ bits.

When performing the mining step using MapReduce, besides the complexity analyzed above, we also need to consider the additional I/O complexity caused by the intermediate results, which is discussed in detail in Section III-C3.

TABLE II
COMPARISONS OF TWO PROBABILITIES P_{occ} AND P'_{occ}

(l, d)	P_{occ} ($p = 0.2$)	P_{occ} ($p = 0.5$)	P_{occ} ($p = 0.8$)	P'_{occ}
(9, 2)	0.65198	0.26929	0.05383	0.00076
(11, 3)	0.52383	0.13715	0.01180	0.00005
(13, 4)	0.42033	0.06948	0.00254	3.0e-6
(15, 5)	0.33700	0.03508	0.00054	1.9e-7
(17, 6)	0.27006	0.01767	0.00011	1.2e-8
(19, 7)	0.21634	0.00889	0.00002	7.3e-10
(21, 8)	0.17326	0.00447	4.9e-6	4.5e-11
(23, 9)	0.13873	0.00224	1.0e-6	2.8e-12
(25, 10)	0.11107	0.00113	2.1e-7	1.8e-13

III. RESULTS AND DISCUSSION

A. Probabilistic Analysis of Mining Emerging Substrings

In this section, we use probabilistic analysis to demonstrate the feasibility of mining emerging substrings for identifying (l, d) motifs, which is the foundation of the MCES algorithm. We expect that the emerging substrings are potential instances of the (l, d) motif. Therefore, we need to demonstrate that i) some instances of the (l, d) motif can exactly occur multiple times in ChIP-seq data sets and ii) they have higher occurrence frequencies than the l-mers in the background sequences.

For this purpose, we compare two probabilities. One is the probability that a random instance of a (l, d) motif occurs (or is implanted) in a given sequence, namely P_{occ} calculated by (1). The other is the probability that a random l-mer occurs in a given sequence of length n, denoted by $P'_{\rm occ}$ calculated by (7). P_{occ} and P'_{occ} represent the occurrence frequencies of a random (l, d) motif instance and that of a random l-mer, respectively.

$$P'_{occ} = \frac{n - l + 1}{4^l}. \quad (7)$$

Table II gives the comparisons of the two probabilities on different (l, d). The values of P_{occ} are obtained in the high, intermediate and low conservation cases by setting p in (2) as 0.2, 0.5 and 0.8, respectively. The values of P'_{occ} are obtained by setting n as 200. From the table, we can find that the values of P_{occ} are large enough in most cases, and thus the random (l, d) motif instances can exactly occur multiple times in the data sets containing thousands or more sequences. Note that, for the cases of large l and p, it is difficult for a (l, d) motif instance to repeat multiple times; in this case, the short segments of this motif instance may repeat multiple times, and we can obtain the motif by mining and clustering these short segments. We can also find that the values of P_{occ} are significantly larger than that of P'_{occ}; that is, random (l, d) instances have higher occurrence frequencies than random l-mers. From the above, it is a feasible way to obtain motif information by mining emerging substrings.

B. Algorithm Assessment

There are two widely used indicators to assess the motif discovery algorithms. One is the running time. The other is the identification accuracy. A good motif discovery algorithm is expected to run in a short time with a high identification accuracy.

We use the nucleotide level performance coefficient (nPC) [29] to evaluate the identification accuracy:

$$nPC = \frac{|K \cap P|}{|K \cup P|}, \quad (8)$$

TABLE III
VALIDITY OF MCES FOR IDENTIFYING (L, D) MOTIFS

(l, d)	Conservation	Running Time	Identificiation Accuracy
(9, 2)	High	5s	1.0
	intermediate	2s	1.0
	Low	2s	1.0
(15, 5)	High	41s	1.0
	intermediate	5s	1.0
	Low	2s	1.0
(21, 8)	High	86s	1.0
	intermediate	14s	1.0
	Low	2s	1.0

TABLE IV
RUNNING TIME ON CHALLENGING (L, D) PROBLEM INSTANCES

(l, d)	MCES	F-motif	TR	TR(OOPS)
(9, 2)	2s	51s	-	5s
(11, 3)	2s	654s	-	7s
(13, 4)	4s	5700s	-	28s
(15, 5)	5s	51396s	-	123s
(17, 6)	8s	-	-	594s
(19, 7)	10s	-	-	2960s
(21, 8)	14s	-	-	14385s
(23, 9)	18s	-	-	61895s
(25, 10)	22s	-	-	-

s: seconds; -: over 24 hours; TR: the TraverStringsR algorithm.

TABLE V
RESULTS ON DATA SETS OF DIFFERENT SIZES

t	MCES		F-motif	
	running time	*memory*	*running time*	*memory*
1000	2s	10M	13331s	313M
1500	9s	16M	20442s	457M
2000	7s	20M	35479s	598M
2500	6s	25M	45899s	727M
3000	2s	24M	51396s	883M
5000	5s	53M	-	-
10000	7s	105M	-	-
50000	29s	526M	-	-
100000	60s	1057M	-	-

s: seconds; -: over 24 hours or out of memory.

where K is the set of nucleotide positions covered by the occurrences of the real motif and P is the set of nucleotide positions covered by the occurrences of the predicted motif. nPC ranges from 0 to 1, indicating both the sensitivity and specificity.

Note that, in our experiments, we simplify the computation of nPC by comparing the real motif and the predicted motif directly: let K and P be the set of nucleotides covered by the real motif and the predicted motif, respectively. That is, assume l_1, l_2 and l_3 be the length of the real motif, the length of the predicted motif and the length of the overlap of the two motifs, respectively; then $|K \cap P| = l_3$ and $|K \cup P| = l_1 + l_2 - l_3$. The reason for using simplified nPC is that it can better reflect the validity of the motif discovery for large input. Let us consider the case that the identified motif and the real motif are identical. In this case, the exact identification is made and the simplified nPC is 1; however, for the original nPC, the spurious hits in searching motif instances in large input will make it very low, which cannot distinguish the identification of different levels effectively.

C. Results on Simulated Data

Simulated data provide quantitative measures to compare the performance of different algorithms. We generate the test set based on the method in [1]: generate a motif of length l and t identically distributed sequences of length n; then, select 80% of the t sequences and implant a random motif instance to a random position in each of the selected sequences. Note that, we generate each motif instance different from the motif in at most d positions, and control the conservation of the (l, d) motif by setting the value of p in (2). The control set consists of t random sequences of length n without implanted motif instances. We implement MCES using C++; all results except those in Section III-C3 are obtained on a computer with a 2.67 GHz processor and a 4 Gbyte memory.

1) Identifying (l, d) Motifs: We first demonstrate the validity of MCES for identifying (l, d) motifs. We choose three (l, d) problem instances of different lengths, and fix $t = 3000$ and $n = 200$. For each problem instance, we test MCES in all the high ($p = 0.2$), intermediate ($p = 0.5$) and low ($p = 0.8$) conservation cases. The results are shown in Table III. MCES not only makes valid identification (outputs the exact results) for all these problem instances but also has quite a short running time. In the case of high conservation, the running time of MCES is obviously larger than the other two cases; the reason is that in this case the mining step mines more emerging substrings and the combining step needs to take more time to process them.

As well as MCES, some other (l, d) motif discovery algorithms can also find the implanted motifs successfully in large data sets, such as F-motif [18]. In this case, the running time is a key indicator used to assess these algorithms. We compare the running time of MCES with two recent and representative (l, d) motif discovery algorithms, namely F-motif [18] and TraverStringsR [11]. F-motif is a more powerful exhaustive method for finding (l, d) motifs in large data sets compared to previous heuristic and exhaustive search algorithms. TraverStringsR is the fastest exact algorithm for identifying (l, d) motifs in dozens of promoter sequences so far. Here, we do not select CisFinder, a fast motif discovery algorithm, as a comparison object, since CisFinder cannot report specified (l, d) motifs.

We show in Table IV the comparisons of running time on different challenging (l, d) problem instances with fixed $t = 3000$ and $n = 200$. MCES runs much faster than the other two algorithms and is able to complete computation on all these instances within one minute. The huge difference of running time is determined by the used algorithm frameworks. For MCES, the mining step is based on the optimal string mining and its running time is fixed for the input data of the same scale; the combining step is based on clustering and its running time depends on the number of mined emerging substrings. The total running time of MCES grows slightly with the increase of l and d, since when handling sequences with large (l, d) motif instances MCES will obtain more emerging substrings and take more time in the combining step. Both F-motif and TraverStringsR are pattern-driven algorithms, and their running time grows dramatically with the increase of l and d. Particularly, TraverStringsR fails to solve any of these problem instances; the reason is that its time complexity under ZOOPS sequence model is $(t - qt + 1)^2$ times that under OOPS sequence model, where q represents the percentage of input sequences containing motif instances [11], and the value of $(t - qt + 1)^2$ is very large for ChIP-seq data sets.

TABLE VI
EFFECT OF THE PARAMETER Λ ON THE PERFORMANCE OF MCES

λ	t = 10000			t = 100000			t = 1000000		
	time	*size*[a]	*num*[b]	*time*	*size*[a]	*num*[b]	*time*	*size*[a]	*num*[b]
w/o	85s	531M	739	691s	4132M	721	4434s	30090M	703
0.1	84s	496M	739	96s	260M	721	206s	255M	703
0.5	30s	50M	703	53s	37M	720	173s	39M	703
0.8	33s	40M	682	49s	27M	692	168s	36M	703

s: seconds; w/o: the case without λ; size[a]: the size of intermediate results; num[b]: the number of mined emerging substrings.

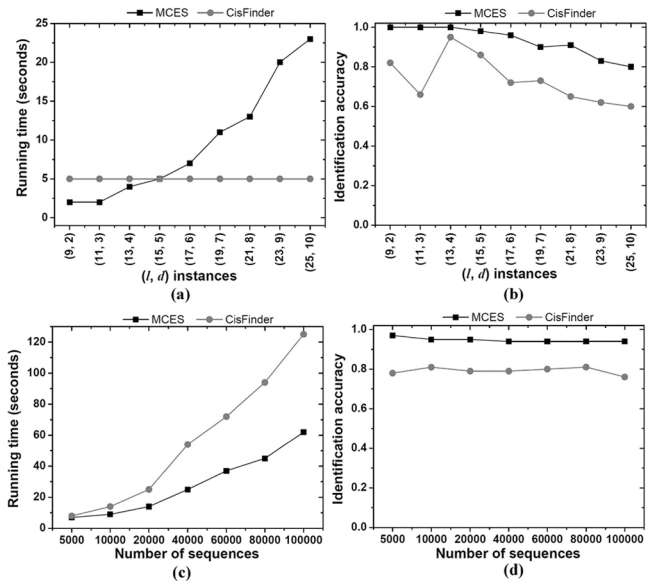

Fig. 2. Comparisons of MCES and CisFinder.

We also list the running time of TraverStringsR under OOPS sequence model and use it as a reference; the results show that, even in this case, TraverStringsR still cannot identify large (l, d) motifs in a practical amount of time.

We show in Table V the comparisons on the data sets of different sizes t with fixed $n = 200$ and $(l, d) = (15, 5)$. MCES is able to identify motifs on all of these data sets and orders of magnitude faster than F-motif. Besides the running time, we also list the required storage space. Although the storage space of MCES and F-motif grows linearly with the data set size t, the storage space of MCES is an order of magnitude smaller than that of F-motif. Our explanation is that MCES and F-motif construct the suffix array and the suffix tree for the input sequences, respectively; the suffix array takes less storage space than the suffix tree.

2) Identifying General Motifs: In this section, we test MCES without giving the length l of the implanted motifs. In reality, the length of the identified motifs is unknown in advance, which increases the problem difficulty, and we call such motifs *general motifs*. We select compared algorithms according to the followings principles: they can analyze full-size ChIP-seq data sets, can complete the calculation in a short time and can make identification without giving the motif length in advance. As a result, we only select CisFinder as the compared algorithm.

First, we carry out comparisons on different (l, d) problem instances with fixed $t = 3000$ and $n = 200$. For each problem instance, the results are the average of ones obtained by running algorithms on three random data sets that correspond to three types of conservation. The running time and identification accuracy are plotted in Figs. 2(a) and 2(b), respectively. Although the running time of MCES grows with the increase of l (this phenomenon has been explained in Section III-C1), it is still small for the maximum l and is competitive with that of CisFinder, a constant running time for fixed t. The identification accuracy of MCES is stable with a slow downward trend and obviously higher than that of CisFinder. We believe this is due to the fact that MCES mines all emerging substrings of different lengths, so as to make use of more information in sequences than CisFinder.

Second, we carry out comparisons on different data set size t (number of sequences) with fixed $n = 200$ and $(l, d) = (15, 5)$. The running time and identification accuracy are plotted in Figs. 2(c) and 2(d), respectively. As our expectations, MCES has a better identification accuracy than CisFinder. For the running time, MCES shows a small upward trend with the increase of the data set size and outperforms CisFinder on the data sets of large size.

3) Identifying Motifs in Larger Data Sets: As shown in Table V, when handling very large data set, the storage space of MCES will exceed the memory capacity of a single machine. We use MapReduce-based strategy to solve this problem. It is interesting to note that, in contrast to our previous work using MapReduce to accelerate motif search in solving difficult problem instances [30], we now use MapReduce mainly to tackle the storage limitation of a single machine in the case of very large inputs.

Our experiments of MapReduce are conducted on a Hadoop cluster with eight personal computers. Each computer has a 2.67 GHz CPU and a 4 Gbyte Memory, and runs the ubuntu 12.04 operating system, JDK1.7 and hadoop 1.2.1. We use the data sets of varied size t and fixed $n = 200$ and $(l, d) = (15, 5)$. We set the size of each data block D_i as 20 Mbytes.

We first test the effect of the parameter λ on the performance of MCES. The parameter λ is used to reduce the intermediate results generated from each map task, so as to improve the time performance of I/O operations. We expect we can filter out some unnecessary substrings and simultaneously retain interesting substrings. Table VI shows the running time, the size of the intermediate results and the number of mined emerging substrings under different values of λ, as well as the case without λ. In the case without λ, we can obtain all emerging substrings safely, but both the running time and the size of the intermediate results are significantly larger than the case using λ, especially for large data set. Also, we can see that although a large λ corresponds to a smaller size of intermediate results and a smaller running time, it may miss some emerging substrings. In order to retain as many emerging substrings as possible, we use a small λ and set it as 0.1 in practice.

We show in Table VII the running time of MCES obtained by varying the number of nodes in the Hadoop cluster. We can find that i) MCES on a single machine without using MapReduce has a good time performance but cannot handle the data set with very

TABLE VII
RUNNING TIME OF MCES USING MAPREDUCE

Number of Nodes	Size of Data Set					
	10000	50000	100000	500000	1000000	5000000
1	7s	29s	60s	-	-	-
2	108s	99s	102s	361s	676s	3228s
4	76s	91s	100s	203s	391s	1594s
8	84s	89s	96s	149s	206s	800s

s: seconds; -: out of memory.

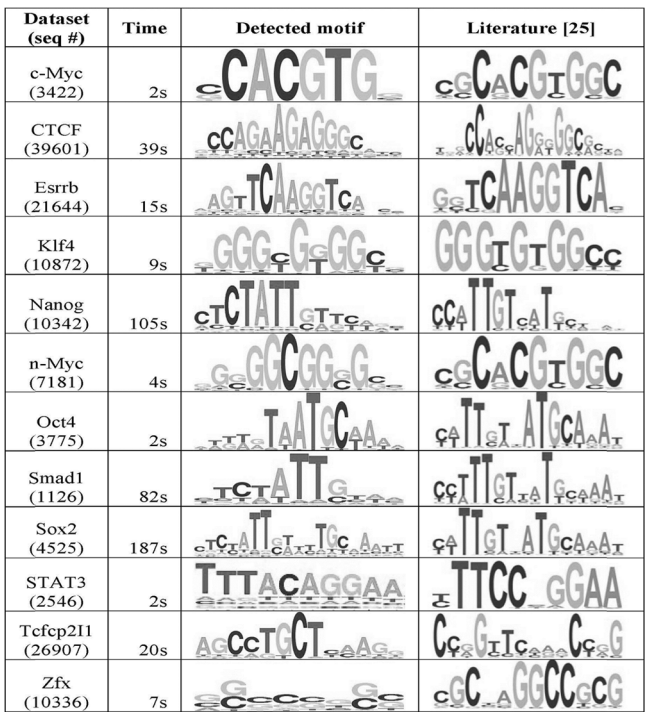

Fig. 3. Results on the mouse embryonic stem cell data sets.

large size; ii) when the data set size is small, the running time on different number of nodes differs slightly, because the total number of partitioned data block is small in this case and not all nodes are used adequately; iii) the advantages of MCES using MapReduce are obvious when the data set size is large and the speedup is almost linearly proportional to the number of nodes.

D. Results on Real Data

Unlike the data sets of promoter sequences, there have been no standard benchmarks for ChIP-seq data sets. Alternatively, we adopt the widely used mESC (mouse embryonic stem cell) data [25], which include 12 ChIP-seq data sets of different sizes for 12 TFBSs. In mESC data, the test sets are composed of peak regions of length 200, and the control sets are composed of background sequences of length 200 starting from 400 base pairs away from both ends of peak regions.

The aim of experiments in this section is to test the validity of MCES for identifying real motifs. Following the previous works using mESC data [18], [20]–[22], we also report the detected motifs in the form of sequence logos, which graphically show the degree of motif conservation measured by relative entropy [31], and then compare them with the motifs published in [25].

We show in Fig. 3 the running time of MCES and the detected motifs. For each of these data sets, MCES is able to find the motif similar to the published one and runs in a short time. For the Nanog, Smad1 and Sox2 data sets, the running time is larger than the simulated data of the same scale; the reason is that MCES obtains a large set of dispersive substrings due to low conservation of motifs in these data sets and needs more time to cluster these dispersive substrings.

It should be pointed out that, besides the published motifs, MCES also reports some new motifs, which are potential DNA binding sites of other transcription factors that bind in complex or in conjunction with the ChIPed transcription factor [14]. In the real world, the transcription factors and the DNA binding sites are not always in the form of one-to-one mapping, and multiple transcription factors may be coupled with multiple binding sites [32], [33]. Multiple conserved sites separated by non-conserved spacers may form a structured motif [34]. Therefore, MCES is helpful for discovering structured motifs in large data sets.

IV. CONCLUSION

We propose a new word counting based algorithm named MCES for identifying motifs in large DNA data sets. MCES offers a new perspective by mining all emerging substrings of different lengths to fully use the information contained in sequences. Experimental results show that MCES is able to efficiently and effectively identify motifs in full-size ChIP-seq data sets.

ACKNOWLEDGMENT

A preliminary version [35] of this work appeared in the proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2–5 November 2014, Belfast, UK. Hongwei Huo is the corresponding author.

REFERENCES

[1] P. A. Pevzner and S. H. Sze, "Combinatorial approaches to finding subtle signals in DNA sequences," in *Proc. ISMB*, 2000, pp. 269–278.
[2] P. A. Evans, A. D. Smith, and H. T. Wareham, "On the complexity of finding common approximate substrings," *Theoretical Comput. Sci.*, vol. 306, pp. 407–430, 2003.
[3] T. L. Bailey and C. Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers," in *Proc. ISMB*, 1994, pp. 28–36.
[4] F. Zambelli, G. Pesole, and G. Pavesi, "Motif discovery and transcription factor binding sites before and after the next-generation sequencing era," *Briefings Bioinformat.*, vol. 14, no. 2, pp. 225–237, 2013.
[5] T. D. Schneider, "Consensus sequence zen," *Appl. Bioinformat.*, vol. 1, pp. 111–119, 2002.
[6] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res.*, vol. 22, pp. 4673–4680, 1994.
[7] Q. Yu, H. Huo, Y. Zhang, and H. Guo, "PairMotif: a new pattern-driven algorithm for planted (l, d) DNA motif search," *PLoS ONE*, vol. 7, no. 10, p. E48442, 2012.
[8] Q. Yu, H. Huo, Y. Zhang, H. Z. Guo, and H. T. Guo, "PairMotif+: a fast and effective algorithm for de novo motif discovery in DNA sequences," *Int. J. Biol. Sci.*, vol. 9, no. 4, pp. 412–424, 2013.
[9] G. Pavesi, G. Mauri, and G. Pesole, "An algorithm for finding signals of unknown length in DNA sequences," *Bioinformatics*, vol. 17, pp. S207–S214, 2001.
[10] H. Dinh, S. Rajasekaran, and J. Davila, "qPMS7: a fast algorithm for finding (l, d)-motifs in DNA and protein sequences," *PLoS ONE*, vol. 7, no. 7, p. E41425, 2012.
[11] S. Tanaka, "Improved exact enumerative algorithms for the planted (l, d)-motif search problem," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 11, no. 2, pp. 361–374, 2014.
[12] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: a gibb's sampling strategy for multiple alignment," *Science*, vol. 262, pp. 208–214, 1993.
[13] K. Wong, T. Chan, C. Peng, Y. Li, and Z. Zhang, "DNA motif elucidation using belief propagation," *Nucleic Acids Res.*, vol. 41, no. 16, p. E153, 2013.

[14] T. L. Bailey, P. Krajewski, I. Ladunga, C. Lefebvre, Q. Li, T. Liu, P. Madrigal, C. Taslim, and J. Zhang, "Practical guidelines for the comprehensive analysis of ChIP-seq data," *PLoS Comput. Biol.*, vol. 9, no. 11, p. E1003326, 2013.
[15] P. Machanick and T. L. Bailey, "MEME-ChIP: motif analysis of large DNA datasets," *Bioinformatics*, vol. 27, no. 12, pp. 1696–1697, 2011.
[16] M. Hu, J. Yu, J. M. Taylor, A. M. Chinnaiyan, and Z. Qin, "On the detection and refinement of transcription factor binding sites using ChIP-seq data," *Nucleic Acids Res.*, vol. 38, no. 7, pp. 2154–2167, 2010.
[17] D. Quang and X. Xie, "EXTREME: an online EM algorithm for motif discovery doi:10.1093/bioinformatics/btu093," *Bioinformatics*, 2014.
[18] C. Jia, M. Carson, Y. Wang, Y. Lin, and H. Lu, "A new exhaustive method and strategy for finding motifs in ChIP-enriched regions," *PLoS ONE*, vol. 9, no. 1, p. E86044, 2014.
[19] F. Zambelli and G. Pavesi, "A faster algorithm for motif finding in sequences from ChIP-seq data," in *Proc. CIBB*, 2011, pp. 201–212.
[20] T. L. Bailey, "DREME: motif discovery in transcription factor ChIPseq data," *Bioinformatics*, vol. 27, no. 12, pp. 1653–1659, 2011.
[21] M. Thomas-Chollier, C. Herrmann, M. Defrance, O. Sand, D. Thieffry, and J. v. Helden, "RSAT peak-motifs: motif analysis in full-size ChIPseq datasets," *Nucleic Acids Res.*, vol. 40, p. E31, 2012.
[22] A. A. Sharov and M. S. Ko, "Exhaustive search for over-represented DNA sequence motifs with CisFinder," *DNA Res.*, vol. 16, pp. 261–273, 2009.
[23] J. Fischer, V. Heun, and S. Kramer, "Optimal string mining under frequency constraints," in *Proc. PKDD*, 2006, pp. 139–150.
[24] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[25] X. Chen, H. Xu, P. Yuan, F. Fang, M. Huss, V. B. Vega, E. Wong, Y. L. Orlov, W. Zhang, J. Jiang, Y. H. Loh, H. C. Yeo, Z. X. Yeo, V. Narang, K. R. Govindarajan, B. Leong, A. Shahab, Y. Ruan, G. Bourque, W. K. Sung, N. D. Clarke, C. L. Wei, and H. H. Ng, "Integration of external signaling pathways with the core transcriptional network in embryonic stem cells," *Cell*, vol. 133, pp. 1106–1117, 2008.
[26] S. van Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, Univ. Utrecht, Utrecht, The Netherlands, 2000.
[27] S. Brohee and J. v. Helden, "Evaluation of clustering algorithms for protein-protein interaction," *BMC Bioinformat.*, vol. 7, p. 488, 2006.
[28] T. Wang and G. D. Stormo, "Combining phylogenetic data with co-regulated genes to identify regulatory motifs," *Bioinformatics*, vol. 19, no. 18, pp. 2369–2380, 2003.
[29] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. D. Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Régnier, N. Simonis, S. Sinha, G. Thijs, J. v. Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu, "Assessing computational tools for the discovery of transcription factor binding sites," *Nature Biotechnol.*, vol. 23, pp. 137–144, 2005.
[30] H. Huo, S. Lin, Q. Yu, Y. Zhang, and V. Stojkovic, "A MapReduce-based algorithm for motif search," in *Proc. IPDPSW*, 2012, pp. 2046–2054.
[31] G. E. Crooks, G. Hon, J. M. Chandonia, and S. E. Brenner, "Weblogo: a sequence logo generator," *Genome Res.*, vol. 14, no. 6, pp. 1188–1190, 2004.
[32] Y. Guo, S. Mahony, and D. K. Gifford, "High resolution genome wide binding event finding and motif discovery reveals transcription factor spatial binding constraints," *PLoS Comput. Biol.*, vol. 8, no. 8, p. E1002638, 2012.
[33] K. Wong, C. Peng, M. Wong, and K. Leung, "Generalizing and learning protein-DNA binding sequence representations by an evolutionary algorithm," *Soft Comput.*, vol. 15, pp. 1631–1642, 2011.
[34] S. Pissis, "MoTeX-II: structured motif extraction from large-scale datasets," *BMC Bioinformat.*, vol. 15, p. 235, 2014.
[35] Q. Yu, H. Huo, X. Chen, H. Guo, J. S. Vitter, and J. Huan, "An efficient motif finding algorithm for large DNA data sets," in *Proc. BIBM*, 2014, pp. 397–402.

Qiang Yu received the B.S. degree, the M.S. degree and the Ph.D. degree from Xidian University, China, in 2006, 2009, and 2014. He is currently working in the Department of Computer Science at Xidian University. His research interests include design and analysis of algorithms, bioinformatics, and parallel and distributed computing.

Hongwei Huo (M'06) received the B.S. degree in mathematics from Northwest University, China, and the M.S. degree in computer science and the Ph.D. degree in electronic engineering from Xidian University, China. She is a Professor and Chair in the Department of Computer Science at Xidian University. Her research interests include the design and analysis of algorithms, bioinformatics algorithms, external memory algorithms and compressed indexes, data compression, parallel and distributed algorithms, algorithm engineering. She is a member of the IEEE Computer Society.

Xiaoyang Chen received the B.S. degree in 2013 from Xidian University, China, where he is currently working toward the Ph.D. degree. His research interests include design and analysis of algorithms, parallel and distributed computing, and graph search.

Haitao Guo received the B.S. degree in 2004 from Anhui Polytechnic University, China, and the M.S. degree in 2007 from Xidian University, China, where he is currently working toward the Ph.D. degree. His research interests include design and analysis of algorithms, and bioinformatics.

Dr. Jeffrey Scott Vitter (F'93) received the B.S. degree with highest honors in mathematics from Notre Dame, South Bend, IN, USA, in 1977; a Ph.D. degree in computer science from Stanford University, Stanford, CA, USA, in 1980; and an M.B.A. degree from Duke University, Durham, NC, USA, in 2002. He is provost and executive vice chancellor and Roy A. Roberts Distinguished Professor at the University of Kansas (KU), Lawrence, KS, USA. He co-led KU's strategic planning and has overseen the first-ever university-wide KU Core curriculum, expansion in engineering and business, multidisciplinary research, major growth of commercialization and corporate partnerships, and administrative efficiency. Since 1980 he has served in administrative leadership and faculty roles at Brown, Duke, Purdue, and Texas A&M. He has over 300 publications, primarily dealing with algorithmic aspects of big data, and is a fellow of the Guggenheim Society, AAAS, and ACM.

Jun Huan (M'11) received his B.S. degree in biochemistry and molecular biology from Peking University, China, in 1997, his M.S. degree in computer science from the Oklahoma State University, Stillwater, OK, USA, in 2000, and his Ph.D. degree in computer science from the University of North Carolina, Chapel Hill, NC, USA, in 2006. Dr. Huan joined the Department of Electrical Engineering and Computer Science at the University of Kansas (KU), Lawrence, KS, USA, in 2006 and is now a professor. At KU Dr. Huan directs the Bioinformatics and Computational Life Sciences Laboratory at KU Information and Telecommunication Technology Center (ITTC). He holds courtesy appointments at the KU Bioinformatics Center, the KU Bioengineering Program, and a visiting professorship from GlaxoSmithKline plc.