BIOINFORMATICS ORIGINAL PAPER
Vol. 24 no. 3 2008, pages 341–349
doi:10.1093/bioinformatics/btm606

Sequence analysis

TFBS identification based on genetic algorithm with combined representations and adaptive post-processing

Tak-Ming Chan*, Kwong-Sak Leung and Kin-Hong Lee

Department of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong

Received on August 8, 2007; revised on November 23, 2007; accepted on December 3, 2007

Advance Access publication December 6, 2007

Associate Editor: Martin Bishop

ABSTRACT

Motivation: Identification of transcription factor binding sites (TFBSs) plays an important role in deciphering the mechanisms of gene regulation. Recently, GAME, a Genetic Algorithm (GA)-based approach with iterative post-processing, has shown superior performance in TFBS identification. However, the basic GA in GAME is not elaborately designed, and may be trapped in local optima in real problems. The feature operators are only applied in the post-processing, but the final performance heavily depends on the GA output. Hence, both effectiveness and efficiency of the overall algorithm can be improved by introducing more advanced representations and novel operators in the GA, as well as designing the post-processing in an adaptive way.

Results: We propose a novel framework GALF-P, consisting of Genetic Algorithm with Local Filtering (GALF) and adaptive post-processing techniques (-P), to achieve both effectiveness and efficiency for TFBS identification. GALF combines the position-led and consensus-led representations used separately in current GAs and employs a novel local filtering operator to get rid of false positives within an individual efficiently during the evolutionary process in the GA. Pre-selection is used to maintain diversity and avoid local optima. Post-processing with adaptive adding and removing is developed to handle general cases with arbitrary numbers of instances per sequence. GALF-P shows superior performance to GAME, MEME, BioProspector and BioOptimizer on synthetic datasets with difficult scenarios and real test datasets. GALF-P is also more robust and reliable when further compared with GAME, the current state-of-the-art approach.

Availability: http://www.cse.cuhk.edu.hk/~tmchan/GALFP/

Contact: tmchan@cse.cuhk.edu.hk

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

Transcription Factor Binding Sites (TFBSs) are small nucleotide fragments (usually ≤ 30 bp) in the *cis*-regulatory regions that can regulate gene expression by interacting with transcription factors (TFs). They are usually 100–3000 bp upstream to the transcription start sites of genes. In higher eukaryotes, they can be found upstream, downstream, or in the introns of the genes that they regulate. Moreover, they can be close to or far away from the regulated genes. TFBSs are a crucial component for gene regulation, which affects the transcription processes and finally the phenotypes of organisms. Typically, when certain TF binds to the TFBS in the promoter region of the corresponding sequence, the transcription process is signaled and initiated. On the other hand, when other competing molecule interacts with the binding site so that the transcription factor fails to bind it, the transcription process will be inhibited. So it is important to identify the TFBSs in DNA sequences for deciphering the mechanism of gene regulation.

Biological experiments such as DNA footprinting (Galas and Schmitz, 1987) and gel electrophoresis (Garner and Revzin, 1981) are the most reliable and accurate methods to identify TFBSs, but they are very expensive and time-consuming. Alternatively, *de novo* TFBS identification or motif discovery, which tries to find out the TFBSs in a computational way without prior knowledge of their consensus appearance, has been proposed thanks to the availability of sequencing and microarray data. The basis is that TFBSs among various organisms for a certain TF are considered similar, sharing some pattern called the consensus or motif. A set of *cis*-regulatory region sequences bound by the same TF can be collected based on their similar expression patterns of the co-expressed genes, and then it is possible to discover the underlying TFBSs by comparing these sequences and extracting the similar small subsequences or fragments from different sequences. We denote these binding sites as motif instances.

However, the challenge lies in the weak conservation due to mutation in evolution, rendering exact matching infeasible. Exhaustive search is prohibitive due to the exponential growth of computation with respect to the increasing problem size. Thus heuristic methods have been proposed. Approximate methods set some constraints (Bieganski *et al*., 1994) and try to provide sub-optimal solutions in polynomial time. However, the approximate solutions contain a large amount of false positives, and as a result further analysis is needed. Multiple sequence alignment methods only provide little insight into the problem, because though similar fragments may be aligned together, various organisms sharing the motif are different in most parts and the positions of the TFBSs are not necessarily proximal. Some machine learning methods such as Expectation Maximization (e.g. MEME; Bailey and Elkan, 1994), Gibbs sampling (Liu *et al*., 1995; Lawrence *et al*., 1993) and Hidden Markov Models (Thijs *et al*., 2001) have been proposed and shown some success.

*To whom correspondence should be addressed.

The potential drawbacks of these methods include that they are sensitive to initial parameter settings, and are often trapped in local optima since many of these methods perform local search only.

Recently, evolutionary computation (EC) methods (Fogel *et al*., 2004), specifically genetic algorithms (GAs) (Liu *et al*., 2004; Che *et al*., 2005; Gertz *et al*., 2005; Paul and Iba, 2006; Stine *et al*., 2003; Wei and Jensen, 2006) have been applied to TFBS identification. GAs are more effective than locally incremental and single-point search methods because GAs perform global search while maintaining a population of different solutions concurrently. The current GA methods employ either position-led or consensus-led representations, respectively, while each type has its own disadvantages. In this article, Genetic Algorithm with Local Filtering (GALF) (see Chan *et al*., 2007 for the preliminary version) is proposed employing the combined representations and a novel local filtering operator to achieve better effectiveness and efficiency. GALF-P, the extension of GALF with adaptive post-processing, is developed to handle more general cases and shows superior performance to the current state-of-the-art approaches.

The rest of this article is arranged as follows: in Section 2, the problem details will be described. In Section Section 3, GALF and GALF-P will be presented in detail. Experimental results will be reported in Section 4, showing the superior and reliable performance of GALF-P. The last section will be the discussion and conclusion.

2 PROBLEM DESCRIPTIONS

2.1 Definitions

Generally, the single TFBS identification problem in unaligned DNA sequences can be defined as two related motif discovery problems corresponding to the position-led and the consensus-led representations, respectively in GAs as follows:

Input: a set of N sequences $S=\{S_1, S_2, \ldots, S_N\}$, each of which is from the finite alphabet $\Sigma(=\{A, T, C, G\}$ for DNA sequences), where the length of each sequence is l, and the motif width w with a valid constraint $0<w\ll l$.

DEFINITION 1. ***General Consensus Patterns*** *(position-led): find a set of instances* $M=\{m_1, m_2, \ldots, m_N\}$ *where each* m_i *is a subsequence with length w from sequence* S_i*, such that the sum of information content IC (proposed by Stormo, 1988)*

$$IC=\sum_{j=1}^{w}\sum_{b} f_b(j)\log\frac{f_b(j)}{p_b} \tag{1}$$

is maximized, where $f_b(j)$ *is the normalized frequency of nucleotide* $b\in\Sigma$ *on the column j of all instances in M and* p_b *is the background frequency of the same nucleotide (from S or the whole genome).*

DEFINITION 2. ***Consensus Pattern*** *(consensus-led): find a string* S_C *with length w from* Σ *and a set of subsequences* $M=\{m_1, m_2, \ldots, m_N\}$ *from S where each* m_i *is with length w from sequence* S_i*, such that the sum of Hamming distances (*d_H*) is minimized*

$$\sum_{i=1}^{N} d_H(S_C, m_i) \tag{2}$$

The equivalent definitions of these two problems were given by Li *et al*. (2002) who have proved both of them to be NP-hard. Definitions 1 and 2 only address a special case of the real single TFBS identification problem, where there can be zero or more than one motif instance according to the motif type in each sequence. This issue will be considered in the following section. There may also be multiple TFBSs corresponding to different types of motifs or consensuses. However, in this article single TFBS identification will be our major concern if not specifically stated.

2.2 Solution space

To analyze the search strategies in GAs, the solution/search space of TFBS identification is discussed here.

For the solution representation in Definition 1 (position-led), different assumptions will lead to different number of instances k_i in S_i according to the previous descriptions. For the most general case where $0\le k_i\le l-w+1$, the solution space is as prohibitively huge as $O((2^{l-w+2})^N)$ (Wei and Jensen, 2006). For the case in Definition 1, where $k_i=1$, the search space is reduced to be $O((l-w+1)^N)$. While allowing $k_i\le 1$, search space becomes $O((l-w+2)^N)$. To make the computation tractable, all the GA approaches are only restricted to the solution space for $k_i=1$ or $k_i\le 1$. We will start with the case of $k_i=1$ which is uniform and widely adopted in GA methods (Che *et al*., 2005; Liu *et al*., 2004; Paul and Iba, 2006). Then more general cases will be addressed by post-processing in later sections.

For the solution representation in Definition 2 (consensus-led), the solution space for all possible consensus strings is 4^w, which is independent of S and M. This representation is less expressive than the one of Definition 1 because the consensus string cannot accurately measure the conservation of nucleotides when they are not fully conserved in the motif.

3 METHODS

The overall framework, namely GALF-P, which consists of the novel Genetic Algorithm with Local Filtering (GALF) and adaptive post-processing techniques (-P), is briefly introduced in Table 1. The details of the framework will be presented in the following subsections.

3.1 GA representations

According to the two previous problem definitions, GA approaches for TFBS identification are categorized into two based on the position-led and consensus-led representations, respectively.

For the position-led representation approaches (Che *et al*., 2005; Wei and Jensen, 2006), each individual is represented by a vector $I=\{p_1, p_2, \ldots, p_N\}$ storing the set of possible starting positions for the TFBS instances in each sequence. I represents a possible solution set $M=\{m_1, m_2, \ldots, m_N\}$ in Definition 1, because each p_i is uniquely mapped to instance m_i with w known. Position-led approaches have more flexibility to move around in the search space because it is free to change any starting position p_i with a random operation, and it is easy to simultaneously change all the positions in an individual. However, the representation cannot provide a detailed view of quality for each TFBS instance because they are evaluated as a whole, and thus cannot distinguish a small portion of unsuitable positions easily.

For the consensus-led representation approaches (Stine *et al*., 2003; Liu *et al*., 2004; Paul and Iba, 2006), each individual is encoded as the

Table 1. The framework of GALF-P. MAXGEN and MAXRUN are the maximal generations of GALF and maximal times to run GALF, respectively

```
i ← 0;
Repeat: i ← i+1; // GALF
  Initialization:
    g ← 0;
    Generate a random population;
  Repeat: g ← g+1
    Random pairing of the population;
    Single-point Crossover;
    Single-point Mutation;
    Local Filtering triggered every 10 generations;
    Shift Operator on the best individual
    when it stagnates for 10 generations;
    Replacement within the pairs;
  Until (g ≥ MAXGEN or Converged)
  Store the so-far-best individual I_Best;
Until (i ≥ MAXRUN)
// Post-processing:
  I_Best+ ← I_Best with adaptive adding;
  I_Best−(Output) ← I_Best+ with adaptive removing;
```

potential consensus in a string pattern $C = c_1c_2...c_w$, which has the same format as S_C in Definition 2. The individuals of consensus-led methods can be generated or extracted randomly from the input sequences. One disadvantage of consensus-led approaches is the computation need to scan all sequences when evaluating a single individual. Furthermore, string patterns are not expressive or accurate enough when different nucleotides of the instances are weakly conserved at some columns of the motif.

3.2 Representations in GALF

Although the two GA representations address TFBS identification differently, they are closely related to each other. For position-led representation, once the optimal I (in other words M) is found, C (S_C) can be easily determined by setting the most frequent letter at the ith column of M as c_i. On the other hand, once the optimal C (S_C) is discovered, the instance set M and I are determined at the same time.

Intuitively it is possible to improve both effectiveness and efficiency by combining the two representations and letting them complement each other with direct refinement on one representation based on the other one. As a result, the position- and consensus-led GALF is proposed. In GALF, the basic representation is based on the position-led one (I) for its flexibility to explore the search space easily. The evaluation function is the information content IC shown in Equation (3), which is similar to Equation (1) except that it only considers the non-zero frequencies.

$$IC = \sum_{j=1}^{w} IC(j) = \sum_{j=1}^{w} \sum_{f_b(j)>0} f_b(j) \log \frac{f_b(j)}{p_b} \tag{3}$$

Meanwhile, the consensus string is not used directly since it is not accurate enough to measure weakly conserved instances. Therefore a Position-specific Weight Matrix (PWM) containing the consensus statistics will be employed to support more accurate measurement (Fig. 1). Each cell in the PWM indicates the normalized frequency of the nucleotide in a particular position of the instance set M. Instead of the Hamming distance d_H in Equation 2 for the string pattern, a more accurate similarity score for evaluating each instance m_i with respect to the PWM can be obtained:

$$Score_{Sim}(m_i) = \sum_{j=1}^{w} f_{m_i(j)}(j) \tag{4}$$

where $m_i(j) \in \Sigma$ is the nucleotide in column j of instance m_i, and $f_{m_i(j)}(j)$ is the corresponding frequency from the PWM. An illustration of the combined representation and the similarity score is shown in Figure 1. For example, $Score_{Sim}(m_2) = 0.2 + 0.2 + 1.0 + 0.8 + 0.8 + 0.6 = 3.6$.

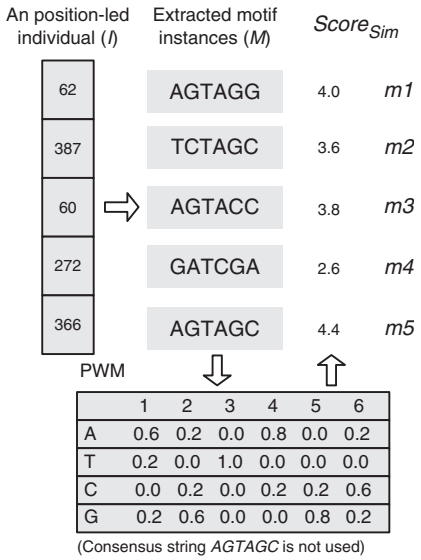

Fig 1. The position-led and consensus-led representations of an artificial individual and the $Score_{Sim}$ of its motif instances calculated from the PWM.

3.3 Local filtering operator

One dilemma of position-led GA approaches is that an individual may be made up of a portion of positions (in other words motif instances) with high similarities between each other, yet another portion is 'false positives' which are poorly aligned to the potential consensus. They cannot be distinguished nor modified efficiently by traditional genetic operators. Consensus-led approaches address this problem by scanning all the sequences each time to evaluate an individual, which imposes heavy computation. Moreover, string representations cannot measure the instances accurately.

With the complementing representations of I and PWM for consensus, the 'false positives' in I can be efficiently filtered out by the novel local filtering operator. Based on $Score_{Sim}$ the local filtering operator scans for best replacements only in the sequences which contain the current worst instances to be filtered out. The procedure is as follows: firstly, the motif instances m_i within an individual is ranked by its $Score_{Sim}(m_i)$. Secondly, the sequence containing the instance with the lowest similarity score is scanned to find the replacing instance (i.e. the corresponding position) with the best $Score_{Sim}$ in that sequence. If the rank does not change, which means the best instance in this sequence is not better than that of the preceding ranked one from another sequence, then the local filtering is stopped. Else the preceding ranked $Score_{Sim}$ now becomes the lowest, and the corresponding sequence containing that instance is selected and scanned as in the first step. This step is repeated until the ranking does not change. Note that the PWM will not be updated in the local filtering for two purposes. One is to save computational load compared with on-line update, and the other is to try not to be too greedy. The pseudo-code is shown in Table 2.

Take the instances from Figure 1 as an example, after ranking the similarity scores, m_4 (2.6) is the worst instance and its preceding ranked instance is m_2 (3.6). So sequence 4 is scanned for the best instance against the consensus. Suppose $AGTAGG$ (4.0) is found, p_4 is updated. Since the score is better than m_2's, the sequence corresponding to m_2

Table 2. Pseudo-code of local filtering operator

```
Input: Individual I = {p_1, p_2, ..., p_N}
Notation: p_i is the starting position of the motif instance m_i in
   Sequence i in I; Score_Sim(m_i) is the similarity score of m_i;
   N is the sequence number.
LOCAL_FILTER(I)
{
   Sort all the instances of I by Score_Sim(·) and obtain the
   order of sequences according to the ranking:
      Rnk(1), Rnk(2), ... Rnk(N);
   //where Score_Sim(m_Rnk(1)) is the highest score and
   //Score_Sim(m_Rnk(N)) is the lowest score
   for (k = N; k ≤ 2, k - -)
   {
      Scan sequence Rnk(k) to get q_Rnk(k) with best Score'_Sim;
      p_Rnk(k) = q_Rnk(k);
      if (Score_Sim(p_Rnk(k)) ≤ Score_Sim(p_Rnk(k-1)))
         Return the new I;
   }
}
```

will be scanned in the next iteration. The iteration goes on until for some sequence, the best instance found is still worse than its preceding ranked one. For example, if the best instance in sequence 2 is not better than m_3, local filtering is stopped.

When an individual is subject to the evolutionary process, only a small number of 'false positives' need to be filtered and only a few sequences need to be scanned. Since this operator is greedy to some degree, in order to keep the contribution of evolutionary process, it is only triggered at the interval of a certain fixed number of generations.

3.4 Evolutionary process

GALF showed better performance compared to different methods including other GAs (Chan *et al*., 2007). With further investigation into the rough fitness landscape of motif discovery, we find it necessary to explore the search thoroughly to locate the global optima. In order to improve GALF, another evolutionary strategy is proposed to achieve more reliable performance. Different from tournament selections (Chan *et al*., 2007; Liu *et al*., 2004; Che *et al*., 2005; Fogel *et al*., 2004; Paul and Iba, 2006; Stine *et al*., 2003; Wei and Jensen, 2006), pre-selection similar to (Mahfoud, 1992) is employed to maintain the diversity in the population.

The evolutionary process is performed in the position-led representation space. All P individuals in the population are randomly partitioned into $P/2$ non-overlapping pairs. In reproduction, each pair of parents Pr_1, Pr_2 are subject to a certain crossover rate, generating two offspring Of_1 and Of_2. Both the offspring and individuals not chosen for crossovers are subject to mutation with certain mutation rate. Single-point mutation (U) and crossover (X) are used. Therefore, there are 4 possible cases, namely, X with U, X without U, U without X and no operation.

For the first two cases with crossovers (X), replacement happens between Pr_1, Pr_2, Of_1 and Of_2. Each parent is paired with the more similar offspring, e.g. Pr_1 with Of_2, based on their Hamming distance. Accordingly Pr_2 is paired with Of_1. In each pair the one with better fitness will survive and replace the other, thus diversity and certain selection pressure are maintained.

For the third case, U without X, a mutant directly replaces its original version. The purpose is to maintain more diversity and variations. In order not to lose the potential optimum, the best-so-far individual is kept and stored separately. Faster convergence may be achieved if selection is applied where the better version replaces the worse one. However, local filtering already does the job when it is triggered, removing small variations of mutation. If such replacement is also performed, the diversity will be significantly decreased and premature convergence may happen.

Shift operator is also applied as it was in Chan *et al*. (2007) to avoid stagnation of the best individual, though the operator will rarely be triggered with a very high variation rate.

3.5 GALF-P with adaptive post-processing

GALF and many other GA approaches (e.g. Fogel *et al*., 2004; Che *et al*., 2005) have the limitation of assuming $k_i = 1$ in each sequence. To further extend GALF, adaptive post-processing is developed to add motif instances and remove false positives, resulting in the GALF-P framework (Table 1). To provide practitioners with more reliable output, in GALF-P, GALF can be run several times (MAXRUN in Table 1) to obtain the overall best individual I_{Best} before the post-processing is performed, similar to the way of GAME.

Post-processing in GALF-P includes adding and removing instances based on the information content IC in Equation (3). IC is widely employed in different TFBS identification approaches and many novel scoring functions serve as generalized extensions of IC (e.g. Jensen and Liu, 2004). Since our focus is on the more effective and efficient search strategy in GAs, we have just adopted IC and more elaborate extensions on problem modeling will be addressed in future work.

Many methods add pseudo-counts to the PWM to avoid the error in computing zero logarithm for unobserved nucleotides when calculating IC in Equation (1). We alleviate this problem by ignoring the $f_b(j)\log f_b(j)/p_b$ term when $f_b(j) = 0$ in Equation (3), similar to the idea that events with zero probability do not contribute to entropy. This strategy works well for GALF assuming one instance per sequence ($k_i = 1$). However, the set of instances we get from GALF based on Equation (3) tend to be the most conserved one, i.e. each instance is the best in terms of fitness among all the instances in the same sequence. In order to accept weaker instances and reject false positives correctly, pseudo-counts are employed in the post-processing to relax the highest conservation from GALF (see Supplementary material for more details). With the best individual I_{Best} output from GALF, its fitness is re-calculated to be $IC'_{I_{Best}}$ including pseudo-counts (1 for each nucleotide at each column).

Both the adding and removing stages of the adaptive post-processing are shown in Table 3. In the adding stage, the goal is to find an additional set M' whose instances on average (δ) increase $IC'_{I_{Best}}$ by more than ε_0, where ε_0 is a small constant value, intuitively proportionate to the motif width w, i.e. $\varepsilon_0 = \beta * w$. ε_0 stands for a minimum non-trivial increase in fitness. In our experiments, β is fixed at a small value 0.001. Since the adding process adjusts δ adaptively, small changes in β do not affect the results. To include certain weaker instances in M', an initial lower bound is also set as $\delta = -\varepsilon_0$. Each time when a temporary M' is created and it does not satisfy our goal of $\delta > \varepsilon_0$, the adaptive lower bound will be set as $IC'_{I_{Best}} + \delta$ based on which a new M' will be created for the next iteration. The stage will converge as long as the maximal increase $\Delta > \varepsilon_0$, which implies there is a non-empty M' with at least one instance to be added. In this case δ is incremented adaptively and definitely will be larger than ε_0 eventually. With M' added to I_{Best} we obtain I_{Best+}.

In the removing stage shown in Table 3, I_{Best-} is initialized as I_{Best+}, so is $IC'_{I_{Best-}}$. A new threshold $\epsilon'_0 = \max(\delta, \epsilon_0 * \gamma, \epsilon_0)$ is set. The maximum between δ and $\varepsilon_0 * \gamma$ intuitively ensures the removal contributes non-trivially to the increase of IC compared to the adding, and ε_0 will be the minimum threshold when $\gamma = 0$. For initialization of the lower bound, $\delta' \leftarrow \epsilon'_0$. The stage iteratively removes the instance with greatest increase Δ' of $IC'_{I_{Best-}}$ among those instances satisfying the threshold criterion $IC'_{I_{Best-}} + \delta'$. If no such instance exists, the removing

Table 3. Pseudo-code of adaptive post-processing

// Adding Stage:
Obtain the best individual $I_{Best} = \{m_1, ..., m_N\}$ output by GALF;
Calculate $IC'_{I_{Best}}$ with pseudo-counts;
$\epsilon_0 \leftarrow \beta * w$; $\delta \leftarrow -\epsilon_0$;
$\Delta \leftarrow \max_{m_{i,k} \notin I_{Best}}(IC'_{+m_{i,k}} - IC'_{I_{Best}})$;
if ($\Delta \leq \epsilon_0$) // Which means $M' = \emptyset$
{ $\gamma \leftarrow 0$; Return $I_{Best+} \leftarrow I_{Best}$; } // Adding stops
while ($\delta \leq \epsilon_0$)
{
$M' \leftarrow \{m_{i,k} | m_{i,k} \neq m_i, IC'_{+m_{i,k}} > IC'_{I_{Best}} + \delta\}$;
$\delta \leftarrow \text{avg}_{m_{i,k} \in M'}(IC'_{+m_{i,k}} - IC'_{I_{Best}})$;
}
$\gamma = |M'|$;
Return $I_{Best+} \leftarrow I_{Best} \bigcup M'$;

// Removing Stage:
$I_{Best-} \leftarrow I_{Best+}$;
$\epsilon'_0 \leftarrow \max(\delta, \beta * w * \gamma, \beta * w)$; $\delta' \leftarrow \epsilon'_0$;
while (1)
{
Calculate $IC'_{I_{Best-}}$ of I_{Best-} with pseudo-counts;
$M' \leftarrow \{m_{i,j} | m_{i,j} \in I_{Best-}, IC'_{-m_{i,j}} > IC'_{I_{Best-}} + \delta'\}$;
if ($M' = \emptyset$)
{ Return I_{Best-}; }
$\Delta' = \max_{m_{i,j} \in M'}(IC'_{-m_{i,j}} - IC'_{Best-})$;
$I_{Best-} \leftarrow I_{Best-} - \{\text{the instance corresponding to } \Delta'\}$;
$\delta' \leftarrow (\epsilon'_0 + \Delta')/2$;
}

$IC'_{+m_{i,k}}$ is the IC if $m_{i,k}$ is added to I_{Best} and $IC'_{-m_{i,j}}$ is the IC if $m_{i,j}$ is removed from I_{Best-}. All IC values are calculated with pseudo-counts.

stage will be ended. In each iteration I_{Best-} and the corresponding $IC'_{I_{Best-}}$ are updated accordingly. The adaptive updating of $\delta' = (\epsilon'_0 + \Delta')/2$ takes into consideration both the current largest fitness increase Δ' and the initial ϵ'_0. Finally I_{Best-} is output as the solution.

The adding stage allows certain weaker instances in M' to be added and at the same time guarantees that the additional set M' on average should contribute positively and non-trivially (more than ε_0) to IC_{Best}. Similarly, the removing stage is stringent so that only the most probable false positives will be removed one by one. The two stages work adaptively to extend GALF for more general cases and refine the solution effectively. Both simulated and real experiments show that the adaptive post-processing is typically effective for identifying additional motif instances and removing false positives.

4 RESULTS

4.1 Parameter setting

The running configurations of GALF are as follows: there are 500 individuals in the population; in the experiments a maximal generation of 300 is shown to be sufficient and the stopping criterion for convergence is that the best individual does not change for 50 consecutive generations; and interval to trigger local filtering is 10 generations. For fair comparisons, we have deliberately set the same number of individuals and convergence criterion as GAME's.

In order to find out the optimal parameter settings for GALF, 54 different combinations of mutation rates (six values: from 0.1 to 0.9 with step 0.2 and 1.0) and crossover rates

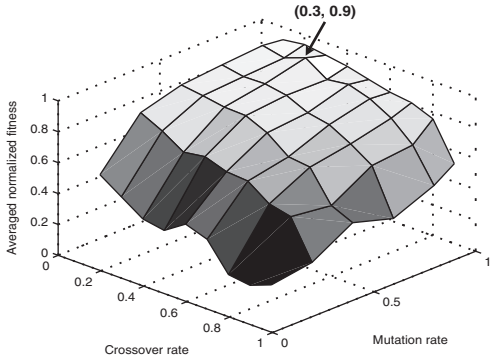

Fig. 2. The normalized fitness averaged on all the datasets for each combination of crossover and mutation rate setting.

(nine values: from 0.1 to 0.9 with step 0.1) are tested for the capability to locate the optimal results. Four synthetic datasets are generated to include different sequence lengths, numbers of sequences, motif widths and different conservation degrees (details in the Supplementary Material). Note that they are totally different from the synthetic datasets experimented in the next section to avoid over-training that may favor our approach. GALF is first run 20 times for each setting on each dataset, and then the average fitness is normalized for different settings. Averaging the normalized fitness for different datasets we have the average normalized fitness for evaluation. Figure 2 shows the averaged normalized fitness for each setting. In general, GALF favors high mutation and moderate crossover rates to keep the diversities that local filtering reduces. The best configuration is 0.9 and 0.3 for mutation and crossover rates, respectively and this setting will be fixed in the following experiments. Since the post-processing in GALF-P only needs the best output from GALF of 20 runs, any setting in the high plateau in Figure 2 is also acceptable, although lower crossover rates need more time to converge.

4.2 Evaluation with synthetic data

In order to evaluate the performance of GALF-P for TFBS identification, a total of 800 synthetic datasets with length 300 bp for each sequence are generated with the following eight combinations of scenarios: (1) motif width: short (8 bp) or long (16 bp); (2) number of sequences: small (20) or large (60); (3) motif conservation: high or low. For each combination, 100 datasets are generated and embedded with the instances of a random motif. In the high conservation scenario, in every column of the motif instances, the dominant nucleotide is generated with 0.91 probability (while all other 3 with 0.03 each). In the low conservation scenario, only 60% of the columns in the motif instances are as highly conserved as in the previous high conservation scenario, while 40% are lowly conserved, where the dominant nucleotide is generated only with 0.55 probability (while all other 3 with 0.15 each) in every instance. To simulate the noisy situation in real data, in each synthetic dataset, the sequences have 10% probability of containing no motif instances. In the rest of them which contain motif instances, there is 10% probability that the sequences have more than one instance. The number of

additional instance(s) in the sequences follows the geometric distribution with $p=0.5$, i.e. $P(k)=(1-p)^{k-1}p$, and therefore $k+1$ instances are embedded in such a sequence.

The performance of GALF-P is compared with GAME, MEME, Bioprospector(BioPro.), BioOptimizers based on MEME and Bioprospector (BioOpt. M. and BioOpt. B., respectively) on the synthetic datasets, with fixed motif widths. The metrics for evaluation are the precision, recall and the *F*-score for information retrieval (Shaw *et al.*, 1997) Precision and recall are defined as follows:

$$Precision = \frac{\#_c}{\#_p}, \text{ and } Recall = \frac{\#_c}{\#_t} \quad (5)$$

where $\#_c$, $\#_p$ and $\#_t$ are the number of correctly predicted motif sites, the number of all the predicted motif sites and the number of all true motif sites embedded in the sequences, respectively. Note that shifting up to 3 bp is allowed for a correctly predicted site, according to (Wei and Jensen, 2006). The *F*-score combining both precision and recall is defined as:

$$F = \frac{2 * Precision * Recall}{Precision + Recall} \quad (6)$$

A high *F*-score indicates both precision and recall are high.

The average results for each combined scenario are shown in Table 4. Best *F*-scores are bolded. Since BioOpt.M. does not improve any result of MEME with respect to the evaluation, only MEME results are shown to save space. GALF-P achieves the best average *F*-score and average recall. GALF-P not only has comparable performance to the best approaches in the relatively easy scenarios (high conservation), but also gives the best results in all difficult ones (low conservation) when other approaches deteriorate significantly and find no true motifs in some datasets (details not shown). These difficult scenarios are more close to the real datasets and the results match well with the real dataset experiments in the next section. Thus we believe that GALF-P is superior to other methods in finding the optimal motifs in more realistic (usually difficult) cases. Note that the assumption of one instance per sequence ($k_i=1$) for GALF is violated in all the synthetic datasets. However, GALF-P can still achieve, respectively, 0.97, 0.99 and 0.98 for average precision, recall and *F*-score in one scenario, demonstrating the adaptive post-processing is effective to tackle the general assumptions in real motif problems.

4.3 Experiments on real datasets

In this section, the one-run results of GALF-P are compared with the reported ones from (Wei and Jensen, 2006) of GAME, MEME, Bioprospector (BioPro.), BioOptimizers based on MEME and Bioprospector (BioOpt. M. and BioOpt. B.) on the eight real datasets tested by Wei and Jensen (2006). The details of the datasets are shown in Table 5. Different ranges of motif widths, numbers of sequences as well as numbers of embedded TFBSs are covered. The evaluation criteria are also the precision, recall and *F*-score. Again up to three shifts are allowed for a correctly predicted site.

The results in terms of *F*-scores are compared in Table 6 (the complete table containing precisions and recalls is in the Supplementary Material). The best results are bolded. GALF-P has the best results in seven out of the eight datasets as well as the overall average. GAME is ranked second best in most of the datasets with one best *F*-score. On average, GALF-P (0.83) and GAME (0.77) give significantly better *F*-scores than the other methods (0.56–0.61). Furthermore, GALF-P achieves the best average precision (0.81), recall (0.87) and *F*-score (0.83) while GAME is the second best in terms of all these three metrics (0.78, 0.77 and 0.77, respectively, see the Supplementary Table).

Table 5. The 8 real datasets

Dataset	CREB	CRP	ERE	E2F	MEF2	MYOD	SRF	TBP
N	17	18	25	25	17	17	20	95
l	200	105	200	200	200	200	200	200
w	8	22	13	11	7	6	10	6
$\#_t$	19	23	25	27	17	21	36	95

N is the number of sequences, l is the sequence length, w is the motif width and $\#_t$ is the number of TFBSs embedded.

Table 4. Average results for the synthetic datasets experiment

Scenario			GALF-P			GAME			MEME			BioPros.			BioOpt.B.		
Width	Num	Con	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Short	Small	Low	0.38	0.56	**0.44**	0.29	0.32	0.30	0.49	0.34	0.39	0.43	0.36	0.39	0.44	0.36	0.39
Short	Large	Low	0.52	0.59	**0.55**	0.42	0.32	0.36	0.63	0.33	0.42	0.58	0.37	0.45	0.59	0.37	0.45
Long	Small	Low	0.87	0.91	**0.89**	0.78	0.87	0.82	0.91	0.86	0.88	0.96	0.74	0.83	0.96	0.74	0.83
Long	Large	Low	0.91	0.90	**0.91**	0.92	0.90	0.90	0.96	0.85	0.90	0.98	0.68	0.80	0.98	0.68	0.80
Short	Small	High	0.73	0.90	0.80	0.71	0.80	0.75	0.87	0.84	**0.85**	0.82	0.75	0.78	0.83	0.76	0.79
Short	Large	High	0.81	0.86	**0.83**	0.83	0.83	**0.83**	0.91	0.76	**0.83**	0.88	0.68	0.76	0.88	0.68	0.76
Long	Small	High	0.97	0.99	**0.98**	0.94	0.99	0.97	0.98	0.99	**0.98**	1.00	0.94	0.97	1.00	0.94	0.97
Long	Large	High	0.97	0.97	0.97	0.98	0.99	**0.98**	0.99	0.98	**0.98**	1.00	0.92	0.96	1.00	0.92	0.96
	Average		0.77	0.84	**0.80**	0.73	0.75	0.74	0.84	0.74	0.78	0.83	0.68	0.74	0.83	0.68	0.74

Width for the motif width, Num for the number of sequences, Con for conservation degree, P for precison, R for recall and F for *F*-score.

4.4 Comparisons between GALF-P and GAME

Since GALF-P and GAME are the best two methods and our focus is on GAs, further experiments are performed to compare GALF-P and GAME. To make a detailed comparison, we run both GALF-P and GAME with fixed motif widths on the same eight datasets, each run 20 times. In each run, the GA procedures, namely GALF, and the GA procedure in GAME are both run 20 times to obtain the best individuals, due to the stochastic nature of GAs, before post-processing is applied. Their best and average performances are compared based on the above metrics.

The best results in terms of *F*-scores (with the associated precisions and recalls) are shown in Table 7. Numeric formats for the corresponding precisions and recalls are shown in parentheses. The better results between GAME and GALF-P are bolded. The precisions, recalls and *F*-scores of GALF are shown as well for reference. Since GALF assumes $k_i = 1$, the recalls and *F*-scores are lower than precisions when there is more than one instance per sequence. However, for those three datasets (labeled with asterisk) satisfying the assumption, the high precisions are identical to the recalls and *F*-scores (note that GALF-P shows comparable results even in these cases when the assumption of GALP is favored). As a result, it is fairer to compare GAME with GALF-P, while the results of GALF provide some information on how the GA part performs independently and how the post-processing in GALF-P extends and improves GALF.

GALF-P gives better precisions compared to GAME in seven out of the eight datasets, thanks to the superior performance of GALF, which achieves very high precisions (0.88 on average). On the other hand, GALF-P obtains comparable recalls (six better than or the same as GAME) among which the optimal recalls for MEF2 and MYOD are obtained. Moreover, GALF-P achieves better precision, recall and *F*-score than GAME averaged over the eight datasets.

For the average performance in 20 runs shown in Table 8, the differences between GAME and GALF-P are even larger. The better results between GAME and GALF-P are bolded. GALF-P achieves better precisions for all but one dataset and better recalls for five datasets. In seven of the eight sets GALF-P obtains better *F*-scores than GAME. As a result, the average precision, recall and *F*-score averaged over the eight sets are all significantly better for GALF-P (by $\geq 20\%$). It implies that GALF-P is more stable and reliable in identifying the TFBSs correctly. We discover that during some runs for datasets CREB, MEF2 and MYOD, GAME was trapped in local optima, indicated by the lower reported fitness values compared with the best ones GAME achieved in the 20 runs. As a result, GAME failed to identify any of the motifs in some runs. This suggests that GAME's GA procedure is not elaborately designed or fully optimized, producing inconsistent results in difficult problems with many local optima. On the other hand, the average results of GALF-P (precision 0.80; recall 0.87; *F*-score 0.82) are consistent and comparable with its best results (precision 0.82; recall 0.88; *F*-score 0.84), demonstrating the robust performance of GALF-P, which is also indicated by the generally smaller SD for CREB, MEF2 and MYOD in Table 8.

Table 6. Comparisons of *F*-scores on the 8 real datasets

Dataset	GALF-P	GAME	BioOpt.M.	BioOpt.B.	MEME	BioPro.
CREB	**0.76**	0.73	0.59	0.67	0.59	0.67
CRP	**0.85**	0.80	0.67	0.67	0.67	0.78
ERE	**0.79**	0.75	0.72	0.75	0.71	0.68
E2F	0.81	**0.90**	0.74	0.70	0.76	0.46
MEF2	**0.97**	0.88	0.88	0.61	0.88	0.71
MYOD	**0.72**	0.48	0.00	0.00	0.00	0.00
SRF	**0.84**	0.80	0.74	0.74	0.67	0.70
TBP	**0.89**	0.84	0.40	0.75	0.36	0.71
Average	**0.83**	0.77	0.59	0.61	0.58	0.56

4.5 Complexity and efficiency

To evaluate the efficiency, we analyze the complexity of the evolutionary process of the GA in GAME and GALF in GALF-P. Suppose there are N sequences, each with the same length l. Motif width is w. Population size is P which is the same

Table 7. Comparisons of GALF-P and GAME on the 8 datasets for 20 runs: Best results (in terms of *F*-scores, together with the corresponding precisions and recalls)

Dataset	GAME			GALF-P			GALF		
	Precision	Recall	*F*-score	Precision	Recall	*F*-score	Precision	Recall	*F*-score
CREB	**14/18 (0.78)**	14/19 (0.74)	**0.76**	16/23 (0.70)	**16/19(0.84)**	**0.76**	13/17 (0.76)	13/19 (0.68)	0.72
CRP	18/21 (0.86)	**18/23 (0.78)**	0.82	**17/17 (1.00)**	17/23 (0.74)	**0.85**	17/18 (0.94)	17/23 (0.74)	0.83
ERE∗	20/38 (0.53)	**20/25 (0.80)**	0.63	**19/23 (0.83)**	19/25 (0.76)	**0.79**	19/25 (0.76)	19/25 (0.76)	0.76
E2F	24/30 (0.80)	**24/27 (0.89)**	0.84	**24/29 (0.83)**	**24/27 (0.89)**	**0.86**	20/25 (0.80)	20/27 (0.74)	0.77
MEF2∗	17/19 (0.89)	**17/17 (1.00)**	0.94	**17/18 (0.94)**	**17/17 (1.00)**	**0.97**	17/17 (1.00)	17/17 (1.00)	1.00
MYOD	10/21 (0.48)	10/21 (0.48)	0.48	**21/37 (0.57)**	**21/21 (1.00)**	**0.72**	15/17 (0.88)	15/21 (0.71)	0.79
SRF	33/45 (0.73)	**33/36** (0.92)	0.81	**33/43 (0.79)**	**33/36 (0.92)**	**0.84**	19/20 (0.95)	19/36 (0.53)	0.68
TBP∗	81/101 (0.80)	81/95 (0.85)	0.83	**86/93 (0.92)**	**86/95 (0.91)**	**0.91**	88/95 (0.93)	88/95 (0.93)	0.93
Average	0.73	0.81	0.76	**0.82**	**0.88**	**0.84**	0.88	0.76	0.81

The results of GALF are shown for reference. Datasets satisfying one instance per sequence are labelled with asterisk.

Table 8. Comparisons of GALF-P and GAME on the 8 datasets for 20 runs: Average results (precisions, recalls and *F*-scores are averaged separately)

Dataset	GAME			GALF-P			GALF		
	Precision	Recall	*F*-score	Precision	Recall	*F*-score	Precision	Recall	*F*-score
CREB	0.43 ± 0.36	0.42 ± 0.36	0.42 ± 0.35	**0.70** ± 0.00	**0.84** ± 0.00	**0.76** ± 0.00	0.76 ± 0.00	0.68 ± 0.00	0.72 ± 0.00
CRP	0.79 ± 0.02	**0.78** ± 0.00	0.78 ± 0.01	**0.99** ± 0.03	0.73 ± 0.02	**0.84** ± 0.03	0.93 ± 0.03	0.73 ± 0.02	0.82 ± 0.03
ERE*	0.52 ± 0.03	**0.78** ± 0.08	0.62 ± 0.05	**0.82** ± 0.01	0.76 ± 0.01	**0.79** ± 0.00	0.76 ± 0.01	0.76 ± 0.01	0.76 ± 0.01
E2F	**0.79** ± 0.02	**0.87** ± 0.02	**0.83** ± 0.02	0.77 ± 0.02	0.85 ± 0.01	0.81 ± 0.01	0.76 ± 0.02	0.70 ± 0.01	0.73 ± 0.02
MEF2*	0.52 ± 0.37	0.55 ± 0.40	0.53 ± 0.37	**0.91** ± 0.09	**0.98** ± 0.08	**0.95** ± 0.09	0.97 ± 0.09	0.97 ± 0.09	0.97 ± 0.09
MYOD	0.14 ± 0.20	0.14 ± 0.19	0.14 ± 0.20	**0.57** ± 0.00	**1.00** ± 0.00	**0.72** ± 0.00	0.88 ± 0.00	0.71 ± 0.00	0.79 ± 0.00
SRF	0.71 ± 0.01	0.86 ± 0.01	0.78 ± 0.01	**0.75** ± 0.03	**0.89** ± 0.06	**0.82** ± 0.05	0.88 ± 0.12	0.49 ± 0.07	0.63 ± 0.09
TBP*	0.81 ± 0.08	0.74 ± 0.11	0.77 ± 0.09	**0.87** ± 0.04	**0.87** ± 0.02	**0.87** ± 0.02	0.88 ± 0.03	0.88 ± 0.03	0.88 ± 0.03
Average	0.59	0.64	0.61	**0.80**	**0.87**	**0.82**	0.85	0.74	0.79

With the ± symbols are the SD. The results of GALF are shown for reference. Datasets satisfying one instance per sequence are labelled with asterisk.

for GALF and GAME. The details of the complexity for GALF and GAME are shown in the Supplementary Material. In summary, the overall complexities for GAME and GALF, respectively are:

$$C_{\mathrm{GAME}} = O(G_1 * P * N * w)$$
$$C_{\mathrm{GALF}} = O(G_2 * P * N * (w + 0.1 * (\log N + l/k)))$$

where 0.1 indicates local filtering is triggered once every 10 generations, 1/k is the averaged percentage of sequences scanned in local filtering, and G_1 and G_2 denote the different maximum generations required in GAME and GALF, respectively.

In fact, C_{GALF} has higher complexity than C_{GAME} when the same generations are used and N and/or l are sufficiently large. However, due to the local filtering, GALF achieves convergence within a maximum $G_2 = 300$ generations in the experiments, while GAME requires $G_1 = 3000$ as the maximum generations. Notice that in real cases, usually $w \geq 5$ and $l \leq 1000$ in the promoter regions. The break even point of $C_{\mathrm{GALF}} > C_{\mathrm{GAME}}$ requires: $N \approx 2^{G_1/(G_2 * 0.1) * w} = 2^{100 * w}$ when quick sort is used in local filtering ($N \approx 100 * w$ even if bubble sort is used), or $l \approx k * w * G_1/(G_2 * 0.1) = 100 * w * k$. k drops significantly according to the real dataset experiments, with the average recorded $k = 4.52$. So it is seldom that $C_{\mathrm{GALF}} > C_{\mathrm{GAME}}$ in the real cases w is ~10–20 and l is within a few thousand bp (usually within 3000 bp) of TFBS identification and thus GALF is usually more efficient than GAME.

However, it is not easy to compare the efficiency between the GA in GAME and GALF. Subject to premature convergence in real problems, the maximal generations may not be used up. Another difficulty is that GAME is implemented in JAVA while GALF-P is implemented in C. Moreover, GAME can only be timed with the GA and post-processing as a whole (and thus we time GALF-P in the same way). The comparison on running time is not a reliable indicator of the efficiency of the algorithms, thus the result quality rather than computing time is the major concern. Nevertheless it can be a reference for the practitioners who have arguments on the slow running time of GAs.

In the previous experiments, GALF-P and GAME are both executed on the same Pentium D 3.00 GHz machine with 1 GB memory, running Windows XP. The details of time comparisons can be found in the Supplementary Material. GALF-P is on average 4.49 times (3.11–10.29 times) faster than GAME. GALF-P and GAME require 61.91s and 291.11s on average, respectively, showing that GAs can provide a reasonable computation solution for the problem.

5 DISCUSSION AND CONCLUSION

As a GA based method for TFBS identification, GAME shows better performance than other approaches. However, the basic GA in GAME is not elaborately designed or fully optimized. In the noisy circumstances for motif discovery in real applications, GAME is likely to be trapped by local optima and the GA results significantly affect the final output in despite of any elaborate post-processing.

In this article, GALF, employing the combined representations associated with a novel local filtering operator and advanced evolutionary process, has been proposed to provide a more effective and efficient GA search algorithm than GAME and other approaches. We have further extended GALF to the GALF-P framework by integrating carefully designed adaptive post-processing. GALF-P gives superior results in the difficult (realistic) synthetic datasets and outperforms GAME in terms of precision, recall and *F*-score averaged on the eight datasets tested in (Wei and Jensen, 2006). Moreover, GALF-P shows more stable and reliable performance than GAME and hence should be favored by practitioners. A recent version of GALF-P is also available to identify instances on both forward and reverse strands.

Further efforts will be put in for several issues, the most important one of which is the fitness function. Since our concern in this article is mainly on improved GA-based searching methods rather than developing a new model for the fitness function, the widely adopted *IC* (also serves as a core part of the Bayesian scoring function for GAME) is employed. Nevertheless, we believe appropriate domain knowledge can be incorporated for a more realistic fitness model. More complete

work on the modeling will be addressed in the future. Another challenging and interesting topic is to design a novel multi-modal GA to discover multiple motifs in a single run, rather than several runs with masking techniques.

ACKNOWLEDGEMENTS

The authors would like to thank the authors of GAME for providing the datasets and details for their experiments, and the anonymous reviewers for their valuable comments. This research is partially supported by the grants from the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK4132/05E and CUHK414107).

Conflict of Interest: none declared.

REFERENCES

Bailey,T.L. and Elkan,C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Menlo Park, CA, pp. 28–36.

Bieganski,P. *et al*. (1994) Generalized suffix trees for biological sequence data: applications and implementations. In *Proceedings of the 27th Hawaii International Conference on Systems Science*. IEEE Press, Wailea, HI, USA, pp. 35–44.

Chan,T.-M. *et al*. (2007) TFBS identification by position- and consensus-led genetic algorithm with local filtering. In *GECCO '07: Proceedings of the 2007 conference on Genetic and evolutionary computation*. ACM, London, England, pp. 377–384.

Che,D. *et al*. (2005) MDGA: motif discovery using a genetic algorithm. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, Washington, DC, USA, pp. 447–452.

Fogel,G.B. *et al*. (2004) Discovery of sequence motifs related to coexpression of genes using evolutionary computation. *Nucleic Acids Res*., **32**, 3826–3835.

Galas,D.J. and Schmitz,A. (1987) DNAse footprinting: a simple method for the detection of protein-DNA binding specificity. *Nucleic Acids Res*., **5**, 3157–3170.

Garner,M.M. and Revzin,A. (1981) A gel electrophoresis method for quantifying the binding of proteins to specific DNA regions: application to components of the escherichia coli lactose operon regulatory system. *Nucleic Acids Res*., **9**, 3047–3060.

Gertz,J. *et al*. (2005) Discovery, validation, and genetic dissection of transcription factor binding sites by comparative and functional genomics. *Genome Res*., **15**, 1145–1152.

Jensen,S.T. and Liu,J.S. (2004) BioOptimizer: a Bayesian scoring function approach to motif discovery. *Bioinformatics*, **20**, 1557–1564.

Lawrence,C.E. *et al*. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.

Li,M. *et al*. (2002) Finding similar regions in many sequences. *J. Comput. Syst. Sci*., **65**, 73–96.

Liu,F.F.M. *et al*. (2004) FMGA: Finding motifs by genetic algorithm. In *BIBE '04: Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering,* IEEE Computer Society, Taichung, Taiwan, pp. 459–466.

Liu,J.S. *et al*. (1995) Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J. Am. Stat. Assoc*., **90**, 1156–1170.

Mahfoud,S.W. (1992) Crowding and preselection revisited. In *Parallel problem solving from nature 2*. North-Holland, Amsterdam, The Netherlands, pp. 27–36.

Paul,T.K. and Iba,H. (2006) Identification of weak motifs in multiple biological sequences using genetic algorithm. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, Seattle, Washington, USA, pp. 271–278.

Shaw,W.M. *et al*. (1997) Performance standards and evaluations in ir test collections: cluster-based retrieval models. *Inf. Process. Manage*., **33**, 144.

Stine,M. *et al*. (2003) Motif discovery in upstream sequences of coordinately expressed genes. In *CEC '03: Evolutionary Computation, The 2003 Congress on,* Vol **3,** IEEE Press, Canberra, Australia, 1596–1603.

Stormo,G.D. (1988) Computer methods for analyzing sequence recognition of nucleic acids. *Annu. Rev. BioChem*., **17**, 241–263.

Thijs,G. *et al*. (2001) A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics*, **17**, 1113–1122.

Wei,Z. and Jensen,S.T. (2006) GAME: detecting cis-regulatory elements using a genetic algorithm. *Bioinformatics*, **22**, 1577–1584.