# Randomized Algorithms for Motif Detection

Lusheng Wang[1], Liang Dong[2], and Hui Fan[3]

[1] Department of Computer Science, City University of Hong Kong
Kowloon, Hong Kong
[2] Department of Computer Science, Peking University, Beijing 100871, P.R. China
[3] School of Information and Electronic Engineering,
Institute of Shandong Business and Technology, Yantai, Shandong, P.R. China
lwang@cs.cityu.edu.hk, dongl@theory.cs.pku.edu.cn, fanlinw@263.net

**Abstract. Motivation:** Motif detection for DNA sequences has many important applications in biological studies, e.g., locating binding sites and regulatory signals, and designing genetic probes etc. In this paper, we propose a randomized algorithm, design an improved EM algorithm and combine them to form a software.
**Results:** (1) We design a randomized algorithm for consensus pattern problem. We can show that with high probability, our randomized algorithm finds a pattern in polynomial time with cost error at most $\epsilon \times l$ for each string, where $l$ is the length of the motif and $\epsilon$ can be any positive number given by the user. (2) We design an improved EM (Expectation Maximization) algorithm that outperforms the original EM algorithm. (3) We develop a software MotifDetector that uses our randomized algorithm to find good seeds and uses the improved EM algorithm to do local search. We compare MotifDetector with Buhler and Tompa's PROJECTION which is considered to be the best known software for motif detection. Simulations show that MotifDetector is slower than PROJECTION when the pattern length is relatively small, and outperforms PROJECTION when the pattern length becomes large.
**Availability:** Free from http://www.cs.cityu.edu.hk/˜lwang/software/motif/index.html, subject to copyright restrictions.

## 1 Introduction

Motif detection for DNA sequences is an important problem in bioinformatics that has many applications in biological studies, e.g., locating binding sites [4], finding conserved regions in unaligned sequences, designing genetic probes [15, 17], etc. *Motif detection problem* can be defined as follows: given $n$ sequences, each is of length $m$, and an integer $l$, where $l \leq m$, find a center string $s$ of length $l$ such that $s$ appears (with some errors) in each of the $n$ given sequences. If no error is allowed, the problem is easy. However, in practice, the occurrence of the center string $s$ in each of the given sequences has mutations and is not exact. The problem becomes extremely hard when errors are allowed. Many mathematic models have been proposed. The following two are important.

R. Fleischer and G. Trippen (Eds.): ISAAC 2004, LNCS 3341, pp. 884–895, 2004.

**The Consensus Pattern Problem:** Given $n$ DNA sequences $\{s_1, s_2, \ldots, s_n\}$, each is of length $m$, and an integer $l$, the *consensus pattern problem* asks to find a center string $s$ of length $l$ and a substring $t_i$ of length $l$ in $s_i$ such that

$$\sum_{i=1}^{n} d(s, t_i)$$

is minimized.

**The Closest Substring Problem:** Given $n$ DNA sequences $\{s_1, s_2, \ldots, s_n\}$, each is of length $m$, and an integer $l$, the *closest substring problem* asks to find a center string $s$ of length $l$ and a substring $t_i$ of length $l$ in $s_i$ such that

$$d = \max_{i=1}^{n} d(s, t_i)$$

is minimized.

$d$ here is called the *radius*. Other measures include the *general consensus score* [8] and SP-score.

Other than mathematic models, motif representation is another important issue. There are three representations, *consensus pattern*, *profile*, and *signature* [7]. Here we focus on consensus patterns and profiles. Let $t_1, t_2, \cdots, t_n$ be $n$ strings of length $l$. Each $t_i$ is an occurrence of a motif. The *consensus pattern* of the $n$ occurrences is obtained by choosing the letter that appears the most in each of the $l$ columns. The *profile* of the $n$ occurrences is a $4 \times l$ matrix $W$, each cell $W(i, j)$ is a number indicating the occurrence rate of letter $i$ in column $j$. Figure 1 gives an example.

```
                      caaccca
                      caacccc
                      catcccg
                      catccct
                      cacccca
                 --------------------
consensus pattern     caaccca
another con. pattern  catccca
                        (a)
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a | 0 | 1 | 0.4 | 0 | 0 | 0 | 0.4 |
| c | 1 | 0 | 0.2 | 1 | 1 | 1 | 0.2 |
| g | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.2 |
| t | 0 | 0 | 0.4 | 0 | 0 | 0 | 0.2 |

(b)

**Fig. 1.** (a) The 5 occurrences of the motif and the consensus patterns. (b) the profile matrix

To evaluate those mathematic models, representations or programs, Pevzner and Sze [16] proposed a challenge problem, which has been studied by Keich and Pevzner [9, 10]. We randomly generate $n(n = 20)$ sequences of length $m(m = 600)$. Given a center string $s$ of length $l$, for each of the $n$ random sequences, we randomly choose $d$ positions for $s$, randomly mutate the $d$ letters from $s$ and implant the mutated copy of $s$ into the random sequence. The problem here is to find the implanted pattern. The pattern thus implanted is called an $(l, d)$-*pattern*. Throughout this paper, we will use this model to do simulations and test our algorithms.

Both the consensus pattern problem and the closest substring problem were proved to be NP-hard and polynomial time approximation schemes have been developed for both problems. However, the closest substring problem seems to be harder than the consensus pattern problem from mathematical point of view. When $l = m$, the consensus pattern problem can be solved in polynomial time, whereas, the closest substring problem becomes the *closest string* problem and still remains NP-hard [11].

In this paper, we adopt the model of consensus pattern. We first design a randomized algorithm for the consensus pattern problem. We show that with high probability, our randomized algorithm finds a pattern in polynomial time with cost error at most $\epsilon \times l$, for any positive number $\epsilon$ given by the user. Then, we propose an improved EM algorithm that outperforms the original EM algorithm. Finally, we develop a software MotifDetector that uses our randomized algorithm to find good seeds and uses the improved EM algorithm to do local search. We compare MotifDetector with Buhler and Tompa's PROJECTION [3] That is considered to be the best known software for motif detection. Simulations show that MotifDetector is slower than PROJECTION when $l$ is relatively small, and outperforms PROJECTION when $l$ becomes large.

The improved EM algorithm itself is an independent contribution. It can greatly enhance the speed and quality of the EM algorithm. It can be embedded into other motif detection software.

## 2 A Randomized Algorithm for Consensus Pattern

For consensus pattern problem, a PTAS was designed in [13, 14]. However, that algorithm is not fast enough to give good solutions in practice. Here we propose a randomized algorithm that works reasonably well in practice.

The technique originates from the PTAS in [14] for the closest substring selection problem. We use it here for the consensus pattern. The idea is to randomly choose $k$ positions among the $l$ positions of the pattern. Then we can guess the true consensus pattern at the $k$ selected positions by trying $4^k$ possible strings of length $k$. The partial consensus pattern (with $k$ guessed letters) is used to search all the given sequences $s_1$, $s_2$, …, $s_n$ to find a $t_i$ from each $s_i$ that is closest to the partial consensus pattern, i.e., the number of mismatches between $t_i$ and the partial consensus pattern at those selected $k$ positions is minimized. Let $K$ be the set of $k$ selected positions, and $t'_i$ and $s$ two strings of length $l$. We use $d(t'_i, s|K)$ to denote the number of mismatches between $t'_i$ and $s$ at the positions in $K$. The algorithm is given in Figure 2.

In Step (3), when using a partial consensus string $s_p$ to search a given string $s_i$, we try to find a substring of length $l$ such that the number of mismatches at those $k$ selected positions is minimized. Steps (1)-(4) generate a candidate of the center string. Since this is a randomized algorithm, different executions generate

1. randomly choose $k$ positions from the $l$ positions for the center string;
2. try all $4^k$ possible (partial) consensus strings;
3. use the partial consensus strings $s_p$ to search all the $n$ given strings and find a substring that is closest to the partial center string from each of the n given strings;
4. reconstruct the center string $s$ based on the $n$ selected substrings of length $l$;
5. repeat 1–4 several times and choose the best result.

**Fig. 2.** A randomized algorithm for consensus pattern

different results. Thus, in the algorithm, we repeat Steps (1)-(4) several times to enhance the quality of the output.

An interesting problem is how big the parameter $k$ should be. $k$ is related to the accuracy of the solution. Let $t_i$ be the true occurrence of the motif in the given string $s_i$ and $t'_i$ be a substring of length $l$ in $s_i$. $s$ denotes the consensus pattern. Assume that $t'_i$ is quite different from $t_i$, say,

$$d(s, t'_i) \geq d(s, t_i) + 2\epsilon l \tag{1}$$

for a small number $\epsilon$ that represents the error rate. $Pr(d(t'_i, s|K) \leq d(s, t_i|K))$ denotes the probability that $d(t'_i, s|K) \leq d(s, t_i|K)$. We will estimate $Pr(d(t'_i, s|K) \leq d(s, t_i|K))$, the probability that $t'_i$ is chosen to be the closest substring to the partial consensus pattern $s|K$. Note that from (1), $d(t'_i, s|K) \leq d(s, t_i|K)$ implies either $d(s, t'_i|K) \leq (d(s, t'_i) - \epsilon l) \times \frac{k}{l}$ or $d(s, t_i|K) \geq (d(s, t_i) + \epsilon l) \times \frac{k}{l}$. Thus, we have

$$\begin{aligned} Pr(d(t'_i, s|K) \leq d(s, t_i|K)) \leq\ & Pr(d(t'_i, s|K) \leq (d(s, t'_i) - \epsilon l) \times \frac{k}{l}) \\ & + Pr(d(s, t_i|K) \geq (d(s, t_i) + \epsilon l) \times \frac{k}{l}). \end{aligned} \tag{2}$$

Note that $d(t'_i, s|K)$ is the sum of $k$ independent random 0-1 variables $\sum_{i=1}^{k} X_i$, where $X_i = 1$ indicating a mismatch between $s$ and $t'_i$ at the $i$-th position in $K$. Thus, from Chernoff's bounds,

$$Pr(d(t'_i, s|K) \leq (d(s, t'_i) - \epsilon l) \times \frac{k}{l}) \leq \exp(-\frac{1}{2}\epsilon^2 k).$$

If $k = \lceil \frac{4}{\epsilon^2} \log(nm) \rceil$, then

$$Pr(d(t'_i, s|K) \leq (d(s, t'_i) - \epsilon l) \times \frac{k}{l}) \leq (nm)^{-2}.$$

Similarly, it can be proved that if $k = \lceil \frac{4}{\epsilon^2} \log(nm) \rceil$,

$$Pr(d(s, t_i|K) \geq (d(s, t_i) + \epsilon l) \times \frac{k}{l}) \leq (nm)^{-\frac{4}{3}}.$$

Therefore,

$$Pr(d(t_i', s|K) \leq d(s, t_i|K)) \leq 2(nm)^{-\frac{4}{3}}.$$

Consider all the $n(m - l + 1)$ substrings of length $l$ in the $n$ given strings, we know that with probability $1 - 2(nm)^{-\frac{1}{3}}$, $\sum_{i=1}^{n} d(s, t_i') \leq \sum_{i=1}^{n}(d(s, t_i) + 2\epsilon l)$. Therefore we have the following theorem.

**Theorem 1.** *Let $k = \lceil \frac{4}{\epsilon^2} \log(nm) \rceil$. With probability $1-2(nm)^{-\frac{1}{3}}$, the algorithm finds a string $s$ of length $l$ and a length $l$ substring $t_i$ for each given string $s_i$ such that $\sum_{i=1}^{n} d(s, t_i') \leq \sum_{i=1}^{n}(d(s, t_i)+2\epsilon l)$. The running time of the algorithm is $O(4^k nml)$.*

Theorem 1 shows that when $k$ is large, we can get a good approximation solution. However, in practice, $k$ has to be a relatively big number in order to get satisfactory results. The speed is far below that of PROJECTION. PROJECTION uses a random projection method to find seeds and uses an EM method to do local search. In the next section, we propose a combined approach that uses the EM algorithm to replace Steps (3) and (4) in Figure 2.

## 3 An Randomized Algorithm Using EM Method

Our method contains two parts. (1) we choose a set of starting points, each a $4 \times l$ weight matrix $W$, representing the initial guess of the motif (using a randomized algorithm). (2) we use the "EM" method to refine the motif. Repeat the above two steps several times and report the best motif found.

### 3.1 Choosing Starting Points

A set of starting points are generated using the following randomized algorithm:

1. Choose $k$ different positions uniformly at random from $l$ positions.
2. For each of the $4^k$ possible strings of length $k$, a matrix $W$ is formed as follows: for column $x$, if position $x$ is among the $k$ selected positions, then set $W(b, x) = 1$, where $b$ is the letter at position $x$, and set the other 3 elements in column $x$ to be 0; otherwise, set $W(b, x) = 0.25$ for $b = A, C, G, T$.

Here we use the profile representation of the motif.

### 3.2 Refining Starting Points with EM Method

Lawrence and Reilly [12] were the first to introduce the Expectation Maximization (EM) algorithm in motif finding problems. Bailey and Elkan [1] used it in multiple motif finding. Buhler and Tompa [3] adopted the EM method in their PROJECTION algorithm in the motif refining step. The following description of the EM algorithm is based on [1].

Let a $4 \times l$ matrix $W$ be the initial guess of the motif. $s_i(j)$ denotes the $j$-th letter in sequence $s_i$. Here is the standard EM algorithm to refine the motif:

1. For each position $j$ in each sequence $s_i$, $s_{ij} = s_i(j)s_i(j+1)\dots s_i(j+l-1)$ denotes the $l$-mer (substring of length $l$) starting at $s_i(j)$ and ending at $s_i(j+l-1)$. Calculate the likelihood that $s_{ij}$ is the occurrence of the motif as follows:

$$P(i,j) = \prod_{x=1}^{l} W(s_{ij}(x), x), \quad 1 \leq i \leq n,\ 1 \leq j \leq m-l+1$$

   where $s_{ij}(x) = s_i(j+x-1)$ is the $x$-th base in $l$-mer $s_{ij}$. In order to avoid zero weights, a fixed small number(we use 0.1) is added to every element of $W$ before calculating the likelihoods.
2. For each $l$-mer $s_{ij}$, we get a normalized probability from the likelihood.

$$P'(i,j) = \frac{P(i,j)}{\sum_{j=1}^{m-l+1} P(i,j)}.$$

   Replace $P(i,j)$ with $P'(i,j)$.
   (The normalization guarantees that $\sum_{j=1}^{m-l+1} P'(i,j) = 1$, reflecting the fact that there is exactly one motif occurrence in each sequence.)
3. Re-estimate the (motif) matrix $W$ from all the $l$-mers as follows:

$$W = \sum_{i=1}^{n} \sum_{j=1}^{m-l+1} W^{ij},$$

   where $W^{ij}$ is also a $4 \times l$ matrix, constructed from $s_{ij}$:

$$W^{ij}(b,x) = \begin{cases} P(i,j) & : \text{ if } b = s_{ij}(x) \\ 0 & : \text{ otherwise.} \end{cases}$$

4. A normalization is applied to $W$ to ensure that the sum of each column in $W$ is 1, i.e.,

$$W'(b,x) = \frac{W(b,x)}{\sum_{b=A,C,G,T} W(b,x)}.$$

   Replace $W$ with $W'$.
5. Steps 1–4 is called a *cycle*. Let $W_{q-1}$ and $W_q$ be the two consecutive matrices produced in cycles $q-1$ and $q$. If

$$\max |W_q(b,x) - W_{q-1}(b,x)| < \epsilon, \tag{3}$$

   then EM stops. Otherwise, goto step 1 and start next cycle.

In step 5, $\epsilon$ is a parameter given by the user. We use a relatively large value $\epsilon = 0.05$ such that on average the EM algorithm stops within very few cycles.

### 3.3 Reporting the Best Motif

For each starting point, at the termination of the EM algorithm, we get $n(m-l+1)$ probabilities $P(i,j)$ standing for the likelihood of $l$-mer $s_{ij}$ being the motif occurrence. We always choose the $s_{ij}$ with the highest $P(i,j)$ as the occurrence of the motif in sequence $s_i$. Buhler and Tompa [3] use the product of the probabilities:

$$Pro = \prod_{i=1}^{n} \max_j P(i,j)$$

to measure the quality of the $t$ occurrences of the motif in the $n$ given sequences. We call $Pro$ the *probability score*. It works well for the easy motifs like (11,2), (13,3) and (15,4). However, for some hard motifs such as (19,6) and (14,4), if $\epsilon$ in (3) is set to be large (0.05), the correct motif often has a lower probability score than some incorrect ones when the EM algorithm stops.

Though setting a smaller value to $\epsilon$ allows the EM algorithm to find the correct motif eventually, the running time of the whole algorithm will be significantly increased. So another choice is to use the measure

$$d = \max_i \min_j d(s_{ij}, c)$$

where $d(,)$ is the hamming distance and $c$ is the consensus string obtained from $W$ by choosing the letter with the biggest probability in each of the $l$ columns. Here for each $s_i$, we select $s_{ij}$ that is the closest to $c$ and we hope that for each $s_i$, $\min_j d(s_{ij}, c) \leq d$ for some given number $d$. An $(l,d)$ motif should have a mismatch-number at most $d$.

### 3.4 The Whole Algorithm

Now we can describe the whole algorithm. See Figure 3.

```
1. for trial = 1 to maxtrials do
2.       choose k positions from l positions uniformly at random;
3.       generate 4^k starting points;
4.       for i = 1 to 4^k do
5.             refine the i-th starting point;
6.             if an (l,d) motif is found then goto 7;
7. report the best motif ever found.
```

**Fig. 3.** The whole algorithm

The algorithm stops as soon as it finds an $(l,d)$ motif. If such a motif can not be found, it stops after *maxtrials* iterations. The parameter $k$ is usually set to be 4.

## 4 Improved EM Algorithm

In this section, we propose some techniques to improve the EM algorithm.

### 4.1 Threshold

The algorithm proposed in Section 3 spends most of the time on running the EM algorithm. Thus, accelerating the EM algorithm is important. In the construction of $W$, those matrices $W^{i,j}$ with very small $P(i,j)$ values represent noise and should be eliminated. Therefore, we use the average value of all $P(i,j)$ for $j = 1, 2, \ldots, m-l+1$, i.e., $\frac{1}{m-l+1}$, as the threshold in Step 3 of the algorithm in Section 3.2. An $l$-mer $s_{ij}$ takes part in the re-estimation of the motif matrix $W$ only when its probability $P(i,j) \geq \frac{1}{m-l+1}$. Thus, Step 3 in Section 3.2 becomes

$$W = \sum_{i=1}^{n} \sum_{j \in G} W^{ij},$$

where $G = \{j \mid 1 \leq j \leq m-l+1, P(i,j) \geq \frac{1}{m-l+1}\}$.

After introducing the threshold, we observe the following improvements:

(1) each EM cycle takes less time than before. Table 1 shows the average running time of every EM cycle for different length of motifs, on a 500 MHz Sun UltraSPARC-IIe workstation. (All the simulations in this paper are done on this computer.)

**Table 1.** Average cycle time (milliseconds). It increases when $l$ increases. The average time for a cycle decreases by about 30%

| $l$ | 11 | 13 | 15 | 17 | 19 |
|---|---|---|---|---|---|
| without threshold | 22.4 | 25.5 | 28.6 | 31.5 | 35.2 |
| with threshold | 16.1 | 18.2 | 20.0 | 22.0 | 23.8 |

(2) EM converges faster. In our experiments, the average number of cycles for the EM algorithm to stop is reduced from 6.0 to 3.6 after using the threshold, which means a time saving of 40%.

(3) the accuracy of the algorithm, i.e., the probability that the EM algorithm finds the correct motif, also increases. Table 2 illustrates the improvement of the accuracy. (Other tables are omitted due to space limit.) In each case, the program runs on 100 random instances. The accuracy and the average running time for different *maxtrials* is reported. Here 20 sequences, each of length 600, are used.

**Table 2.** (17,5)-motif

| | *maxtrials* | 10 | 20 | 30 |
|---|---|---|---|---|
| accuracy (%) | without threshold | 28% | 48% | 60% |
| | with threshold | 59% | 82% | 89% |
| average time (seconds) | without threshold | 423 | 720 | 921 |
| | with threshold | 119 | 161 | 195 |

From these tables we can see that adding the threshold can improve both the accuracy and the speed of the algorithm. The improvement on speed is more significant when accuracy requirement increases since the EM algorithm with threshold can find the correct motif earlier and thus stops earlier.

## 4.2 Shifting

We have often observed the following phenomenon in experiments:

```
tgtgggtcacc   is the actual motif consensus;
ctgtgggtcac   is the motif consensus found by the algorithm.
```

The algorithm spend some time on finding good, but not the correct motif. Those detected patterns have long overlaps with the correct motif. Thus, we can shift the detected motifs with good scores to the left and right for a few positions and see if there is any improvement.

Suppose a motif and its $n$ occurrences are found. Given a number $h$ for shifting, a new motif is produced in this way:

1. For each occurrence $s_{ij}$ of the motif, replace it with $s_{i(j+h)}$.
2. Construct the matrix $W$ based on the new occurrences, and refine it with the EM algorithm.

In our experiment, $h$ is set to be $\pm 1$ and $\pm 2$. The original motif is treated as $h = 0$.

The algorithm is given in Figure 4.

1. for *trial* $= 1$ to *maxtrials* do
2.     choose $k$ positions from $l$ positions uniformly at random;
3.     generate $4^k$ starting points;
4.     for $i = 1$ to $4^k$ do
5.         refine the $i$-th starting point;
6.         if an $(l, d)$ motif is found then goto 9;
7.     shift the top 10 results in this trial;
8.     if an $(l, d)$ motif is found then goto 9;
9. report the best motif ever found.

**Fig. 4.** The whole algorithm with shifting

Table 3 illustrates the improvement using the shifting technique. (Other tables are omitted due to space limit.)

**Table 3.** (15,4)-motif

| | *maxtrials* | 2 | 4 | 8 |
|---|---|---|---|---|
| accuracy (%) | without shift | 27% | 43% | 68% |
| | with shift | 70% | 94% | 100% |
| average time (seconds) | without shift | 29 | 50 | 78 |
| | with shift | 27 | 34 | 36 |

We can see that the shifting technique can improve both the accuracy and the speed of the algorithm. The improvement on speed is more significant on a large *maxtrials*. On a small *maxtrials* the shifting technique sometimes even increases the running time a little bit.

## 5 Implementation of the Software

We put all the techniques together and produce a software, MotifDetector. The program is written in Java 1.1. MotifDetector allows the users to input their own data. The sequences should be in FASTA format. In other words, each sequence has a title line starting with ">" followed by one or more characters (as the title of the sequence) and terminated with an end-of-line. The following lines are assumed to be the sequence until an end-of-file, a blank line, or another line beginning with ">" is encountered. The users can either directly type the sequences in the input area or copy the data from a file and paste them to the input area.

## 6 Comparison with PROJECTION

In [3], Buhler and Tompa developed a software, PROJECTION, using a random projection algorithm. To our knowledge, this is the best known software for motif detection. PROJECTION can solve the (15,4)-motif challenge problem efficiently, and it can even solve (14,4)-motif, given plenty of time. Figure 5 is an outline of the random projection algorithm.

```
1. for trial = 1 to maxtrials do
2.     choose k positions from l positions uniformly at random;
3.     hash all the l-mers of the input sequences into 4^k buckets;
4.     pick up those buckets that receive at least s l-mers;
5.     refine those buckets using EM;
6.     if an (l,d) motif is found then goto 7;
7. report the best motif ever found.
```

**Fig. 5.** The random projection algorithm

Simulations on various cases have been done to compare our software with PROJECTION. The parameters for PROJECTION are set as in [3] whenever possible. PROJECTION's *synthetic mode* is enabled, which improved its performance noticeably. See Table 4 for the result.

From Table 4 we can see: (1) PROJECTION runs faster on (11,2), (13,3), (15,4), (10,2), (12,3), (9,1) and (8,1) motifs; (2)our program is both faster and more accurate on the other 6 cases. It seems that PROJECTION is more efficient on short motifs, and our program is good at solving long and subtle motifs.

**Table 4.** Performance comparison with PROJECTION. On each motif problem, both programs take the same 100 random instances as input. Each problem instance consists of 20 sequences each of length 600. For PROJECTION, set $k = 7$ and $s = 4$. For our program, set $k = 4$, except for (18,6)-motif, on which we found $k = 5$ is better. Running time (in seconds) is averaged on all the 100 instances

| | PROJECTION | | | our program | | | |
|---|---|---|---|---|---|---|---|
| motif | *maxtrials* | accuracy | time | $k$ | *maxtrials* | accuracy | time |
| (11, 2) | 6 | 100% | 5.6 | 4 | 3 | 100% | 13 |
| (13, 3) | 12 | 100% | 12 | 4 | 4 | 100% | 21 |
| (15, 4) | 30 | 100% | 26 | 4 | 8 | 100% | 36 |
| (10, 2) | 60 | 100% | 25 | 4 | 30 | 100% | 47 |
| (12, 3) | 300 | 99% | 203 | 4 | 70 | 99% | 207 |
| (17, 5) | 160 | 99% | 83 | 4 | 16 | 100% | 52 |
| (19, 6) | 200 | 99% | 194 | 4 | 30 | 100% | 92 |
| (14, 4) | 800 | 89% | 936 | 4 | 160 | 90% | 813 |
| (16, 5) | 1200 | 76% | 2334 | 4 | 300 | 82% | 2063 |
| (18, 6) | 2400 | 81% | 4929 | 5 | 150 | 89% | 4661 |
| (9, 1) | 1 | 100% | 4.3 | 4 | 1 | 100% | 6.9 |
| (8, 1) | 3 | 100% | 4.3 | 4 | 5 | 100% | 9.6 |
| (21, 7) | 400 | 96% | 332 | 4 | 60 | 100% | 102 |

## 7 Conclusions

We have developed a software MotifDetector for motif detection. Simulations show that MotifDetector is slower than PROJECTION when the pattern length is relatively small, and outperforms PROJECTION when the pattern length becomes large.

Another contribution is an improved EM algorithm. It can be applied in many current algorithms such as MEME and PROJECTION. We can expect that after using the improved EM algorithm, their performances might be significantly improved.

## Acknowledgement

The work is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 1047/01E].

## References

1. Bailey,T. and Elkan,C. (1995) Unsupervised learning of multiple motifs in biopolymers using expectation maximization, *Machine Learning*, 21, pp.51–80.

2. Blanchette,M. (2001) Algorithms for phylogenetic footprinting. *RECOMB 01: Proceedings of the Fifth Annual International Conference on Computational Molecular Biology*, pp.49–58..
3. Buhler,J. and Tompa,M. (2002) Finding motifs using random projections. *Journal of Computational Biology*, 9, pp.225–242.
4. Dopazo,J., Rodríguez,A., Sáiz,J.C. and Sobrino,F. (1993) Design of primers for PCR amplification of highly variable genomes, *CABIOS*, 9, pp.123–125.
5. Duret,L. and Bucher,P. (1997) Searching for regulatory elements in human noncoding sequences. *Curr. Opin. Struct. Biol.*, 7, pp.399–406.
6. Duret,L., Dorkeld,F. and Gautier,C. (1993) Strong conservation of non-coding sequences during vertebrates evolution: potential involvement in post-transcriptional regulation of gene expression. *Nucleic Acids Research*, 21, pp.2315–2322.
7. Gusfield, D. (1997) Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, *Cambridge University Press*.
8. Hertz,G. and Stormo,G. (1995) Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps, *Proc. 3rd Int'l Conf. Bioinformatics and Genome Research*, pp.201–216.
9. Keich,U. and Pevzner,P. (2002a) Finding motifs in the twilight zone. *Bioinformatics*, 18, pp.1374–1381.
10. Keich,U. and Pevzner,P. (2002b) Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics*, 18, pp.1382–1390.
11. Lanctot,K., Li,M., Ma,B., Wang,S. and Zhang,L. (1999) Distinguishing string selection problems, *Proc. 10th ACM-SIAM Symp. on Discrete Algorithms*, pp.633–642. Also to appear in *Information and Computation*.
12. Lawrence,C. and Reilly,A. (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences, *Proteins*, 7, pp.41–51.
13. Li,M., Ma,B. and Wang,L. (2002) Finding Similar Regions in Many Sequences, *J. Comput. Syst. Sci.*, 65, pp.73–96, special issue for *Thirty-first Annual ACM Symposium on Theory of Computing*.
14. Li,M., Ma,B. and Wang,L. (2002) On the closest string and substring problems, JACM, 49(2): pp.157–171.
15. Lucas,K., Busch,M., Mössinger,S. and Thompson,J.A. (1991) An improved microcomputer program for finding gene- or gene family-specific oligonucleotides suitable as primers for polymerase chain reactions or as probes, *CABIOS*, 7, pp.525–529.
16. Pevzner,P. and Sze,S. (2000) Combinatorial approaches to finding subtle signals in DNA sequences. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*. pp.269–278.
17. Proutski,V. and Holme,E.C. (1996) Primer Master: a new program for the design and analysis of PCR primers, *CABIOS*, 12, pp.253–255.