Primer Set Selection in Multiple PCR Experiments [*]

Wei-Ting Liu, Chang-Biau Yang[†] Shyue-Horng Shiau
Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan

Yow-Ling Shiue
Institute of Biomedical Science
National Sun Yat-sen University, Kaohsiung, Taiwan

Abstract

The selection of a suitable set of primers is crucial to the multiple PCR (*polymerase chain reaction*) experiment, which is one of the most important techniques in molecular biology. The minimum primer set (MPS) problem is to minimize the number of primers required to amplify a set of DNA sequences, so that the experimental costs and time will be reduced. However, the MPS problem has been proved to be NP-complete. In this paper, we propose an efficient heuristic algorithm for solving the MPS problem. In our algorithm, the kernel procedure is to choose a set of possible good primer candidates, each may be able to amplify more than one target DNA sequence. The kernel procedure is accomplished by the local motif finding method, which is based on the combination of the Gibbs sampler method, the ant colony optimization (ACO) strategy, and Liao's algorithm for finding motifs. We add a new *weight* parameter to the method, which can guide us to find local motifs with the local view. Then, the complementary sequences of those local motifs (possible primer candidates) are input into the binary integer programming for getting the near optimal set. With the concept of possible primer candidates, the size of the solution space in the binary integer programming can be reduced drastically. We perform experiments on some artificial domains and two gene families. The experimental results show that the time required for our algorithm is reduced drastically, while the performance of our algorithm is comparable to that obtained by the exhaustive search.

[*]This research work was partially supported by the National Science Council of the Republic of China under NSC-93-2213-E-110-004.

[†]To whom all correspondence should be sent: cbyang@cse.nsysu.edu.tw

Key words: bioinformatics, algorithm, primer, multiple PCR, local motif

1 Introduction

The polymerase chain reaction (PCR) is a method for amplifying DNA sequences fast and massively. With PCR, it is possible to increase the amount of a given DNA sequence many times. The PCR technique has made a significant revolution in molecular biology and genetic analysis.

A *primer* is a short nucleotide chain to which DNA polymerase can add nucleotides. So a primer can be defined as a short string pattern over alphabet= $\{A, G, C, T\}$ [19]. The goal of PCR is to amplify the specific DNA target fragment between a pair of primers, the *forward primer* and the *reverse primer*. A forward primer is suitable for a DNA target fragment if its complementary sequence is a substring in the DNA forward region. And, a reverse primer is suitable for a DNA target fragment if its complementary sequence is a substring in the DNA reverse region. There are three major steps included in PCR, which are repeated $30 \sim 40$ cycles. They are denature step, annealing step and extension step. In this paper, a primer is assumed to be able to amplify a DNA sequence if the primer's complementary sequence is the substring of the DNA sequence, with allowing either exact match or some mismatches.

Many experiments require PCR amplifications of many different targets. It is possible to assemble these PCR amplifications in a single experiment, called *multiple PCR*. Therefore it will save money and time. In multiple PCR, a primer that is designed to amplify more than one target sequence is called a *universal primer* [7].

The selection of a suitable set of primers is very important for PCR. Most existing algorithms for primer selection emphasize in producing the primer pair for each target fragment. In most previous study on the primer selection, the size of the primer set was not considered. Kampke et al. [9] used dynamic programming to design primers in multiple PCR experiments. Their criteria for selecting the best primer set are the traditional criteria such as melting temperatures, GC-content, length, 3' termini, and self-complementary sequences.

Nicodeme et al. [13] showed that the multiple primer selection problem is NP-complete, and designed an efficient approximation algorithm for solving it. Linhart et al. [12] solved the problem, which is given a set of strings with length m and an integer constant d, they were asked to find a primer of length m which matches a maximum number of input strings with degeneracy at most d.

In this paper, we are concerned about the *minimum primer set* (MPS) problem, which is requested to select a minimum set of primers to amplify all given DNA sequence targets simultaneously. In other words, each of given DNA sequences has to be *covered* by at least one primer in the primer set. Pearson et al. [14] proved that the MPS problem is NP-complete. They proposed a branch-and-bound algorithm to solve the problem. In 1996, Pearson et al. [15] proposed another two heuristic algorithms for solving the MPS problem. Doi [4] used a greedy method to produce primers, in which some basic constraints were considered. Talaat et al. [18] also proposed another greedy method.

In recent years, no researcher has ever been devoted to the study of finding the accurate solution for the MPS problem. Hsieh et al. [7] proposed a new point in the MPS problem. They moved the sliding window one base at a time to find all possible primer candidates. And then they formulated the MPS problem as a minimization problem solved by *binary integer programming* (BIP) [6]. But their method produces too many primer candidates (decision variables) in BIP. In this paper, we propose an algorithm for solving the MPS problem based on the combination of the Gibbs sampler method [10], the ant colony optimization (ACO) strategy [5], and Liao's algorithm for finding motifs [11]. Our algorithm can find the initial possible good primer candidates efficiently, and the size of the primer candidate set is reduced. Thus it helps us to reduce the execution time and decision variables in BIP.

In the past few years, several articles have been devoted to the study of designing a pair or multiple pairs of primers for the specific target region of given sequences. In this paper, we do not care about the location where the primer anneals the sequence. We want to make sure whether the primer anneals the sequence or not. With our method, we can find or detect some characteristics of gene families. So it will help biologists to determine beneficial or pathogenic genes in gene families.

The rest of this paper is organized as follows. In Section 2, we will introduce some previous related algorithms, including the Gibbs sampler method, the ACO algorithm, Liao's algorithm for sequence motif finding, and Hsieh's for MPS. We shall propose an algorithm for solving the MPS problem in Section 3. Then an example will be given to illustrate our algorithm in Section 4. In Section 5, we shall show our experimental results on real and artificial domains. And finally, Section 6 concludes this paper.

2 Previous Related Algorithms

The goal of local multiple sequence alignment (MSA) is to locate short patterns shared by all given sequences, which are well known as *motifs*. The detailed characteristics of motifs can be found in published papers [8]. That is, the local MSA can be used for the motif finding problem. The algorithms include the expectation maximization algorithm [3, 17], Gibbs sampler method [10], and Liao's algorithm [11]. In this section, we shall review some algorithms which will be applied by our algorithm for solving the MPS problem.

2.1 The Gibbs Sampler Method

Lawrence et al. [10] proposed a local MSA algorithm to find motifs in n sequences. They used a probabilistic matrix to describe a common pattern, and the search approach is based on the random iterative sampling.

The algorithm maintains two main data structures, a probabilistic model and a set of positions in the given sequences. The probabilistic model is a two-component mixture model. One, denoted as Q, is use to calculate the probability of amino acid (or nucleotide) frequencies for each position of a set of subsequences, each with length w. The other, denoted as D, is the *background* probability that each amino acid (or nucleotide) occurs in all given sequences. The second data structure consists of a set of positions a_i, $1 \leq i \leq n$, where a_i denotes the starting position of pattern x_i in sequence i.

Let P_z be a substring of S^* with length w and starting at position z. For each starting position of a substring with length w, its kernel is to calculate a score q_z, which includes the frequency matrix Q and the background distribution D as follows. [10,11,16].

$$q_z = \sum_{j=1}^{w} \sum_{r=1}^{20} Q_{r,j} log \frac{Q_{r,j}}{D_r} \tag{1}$$

With the score of q_z, the method determine the goodness of that position. If it is a good position, then it is will be a good sampler.

2.2 The Ant Colony Optimization Algorithm

The ant colony optimization (ACO) algorithm is enlightened by the behavior of real ant colony. The main idea of the ACO algorithm is to exchange information within the colony of ants. In the real world, ants communicate with each other via pheromone. In ACO, the ants can be regarded as artificial ants, and the numeric information representing artificial pheromone trails is updated by the behavior of artificial ants. Dorigo et al. [5] proposed an ant colony optimization algorithm to solve NP-hard problems such as the traveling salesman problem. The ACO algorithm is as follows.

Step 1: Set all parameters and initialize pheromone trails.

Step 2: Every ant produces a solution (Suppose there are m ants.).

Step 3: Calculate the scores of the m solutions.

Step 4: Update the sum of pheromone trails.

Step 5: If the solution has not been improved after some predefined iterations, terminate the algorithm; otherwise repeat the iteration (go to Step 2).

2.3 Liao's Algorithm for Sequence Motif Finding

Liao et al. [11] proposed an algorithm for sequence motif finding, which combines ACO with the Gibbs sampler method. In the Gibbs sampler method, it randomly chooses a beginning position a_i of each sequence S_i in the input set, so it is very uncertain to tend to a better solution, and it may lead the solution to be locally optimal.

Liao's algorithm is to apply ACO to find a set of good candidate positions in every sequence. And then input those candidate positions into the Gibbs sampler method. So this method only calculates the scores of the candidate positions instead of all positions in the sequences. It takes less time than the Gibbs sampler method, and the result is even better than the Gibbs sampler method, since the latter method with random beginning positions is very easy to fall into the local optimal trap [11].

Liao's algorithm rewrote the formulas of the pheromone probability and the pheromone update in ACO as follows [11].

Probability: At first, every ant produces a solution by choosing a character in every position. Ant h will choose character c at position z according to the pheromone probability selection rule. The pheromone probability is given as follows.

$$p_h(z_c) = \frac{[\tau_{z_c}(t)]^\alpha}{\sum_{c \in C} [\tau_{z_c}(t)]^\alpha} \tag{2}$$

In Formula 2, $p_h(z_c)$ means the probability that ant h chooses character c at position z. α is a parameter that is used to control the influence of the pheromone trails, and C is the character set of the given sequences. $\tau_{z_c}(t)$ is the pheromone update formula.

Pheromone update: After all ants have produced their solutions, the pheromone will be updated. The consistency of pheromone on each node is reduced by a constant parameter. And then every ant will add its pheromone on the nodes it has passed. The way of the pheromone update is given as follows.

$$\tau_{z_c}(t+1) = (1-\rho) \cdot \tau_{z_c}(t) + \sum_{h=1}^{m_{z_c}} \triangle \tau_{z_c}^{h}(t) \tag{3}$$

In Formula 3, $\tau_{z_c}(t+1)$ means the consistency of pheromone on character c at position z. The parameter ρ, $0 < \rho < 1$, denotes the rate of the pheromone trails evaporation. m_{z_c} denotes the number of all ants which choose character c at position z. $\triangle \tau_{z_c}^{h}(t)$ denotes the variable of pheromone on character c at position z that ant h has selected.

2.4 Hsieh's Method

Hsieh et al. [7] proposed an algorithm for solving the MPS problem, which is given a set I of n DNA sequences $S_1, S_2, \cdots, S_n$. They considered only the forward primer set selection problem. First, they use an exhaustive search method to find a set P_f of all possible primer candidates by scanning all given targets with a fixed size of sliding window. And then they transform the set as the input of binary integer programming. The P_f selection can be stated as follows [7].

$Minimize\ |P_f|$
subject to:
Each sequence S_i in I can be amplified by at least one primer in P_f

They use matrix C_f to represent the amplification ability of primers in P_f as follows.

$$C_f(i,j) = \begin{cases} 1, & \text{if } P_f(j) \text{ can amplify sequence } S_i; \\ 0, & \text{otherwise.} \end{cases}$$

where $P_f(j)$ represents the jth primer in P_f, $1 \le j \le |P_f|$.

The decision variable y_j is involved as follows.

$$y_j = \begin{cases} 1, & \text{if } P_f(j) \text{ is chosen;} \\ 0, & \text{otherwise.} \end{cases}$$

Now the minimization problem can be formulated as binary integer programming:

$$Minimize\ \sum_{j=1}^{|P_f|} y_j$$
$$\text{subject to } C_f Y \ge 1_n,\ 1 \le j \le |P_f|,\ y_j\text{=0 or 1,}$$

where $Y = (y_1, y_2, \cdots, y_{|P_f|})^t$, t means the transpose operation in the matrix , and 1_n represents the $n \times 1$ column vector with all elements equal to 1.

3 Our Algorithm for Local Motif Finding

We use the concept of *local motifs* to find the good possible universal primer candidates. The *local motifs* are common short patterns that are shared by at least two given sequences (allowing either exact match or some mismatches), but may not be shared in all sequences. So it is unnecessary that the pattern appears in every given sequence. According to characteristics of the local motif, the universal primer finding can be viewed as the local motif finding.

In order to find local motifs, we modify the score function in the probabilistic model of the Gibbs sampler method [10]. We bring up a new *weight* parameter, $weight(i)$, which represents the weight of sequence S_i. The heavier weight of a sequence, the more contribution of it for determining the next local motif. If a sequence has been covered by some local motifs found in previous iterations, its weight is decreased. The *weight* function is defined as follows.

$$weight(i) = \frac{1}{same(i)+1}\ , \tag{4}$$

where $same(i)$ denotes the total number of sequences, including sequence S_i itself, that contain the same local motif with sequence S_i.

For every substring x_i of length w, we create a 0/1 probability matrix $A_i(r,j)$ with size $|C| \times w$. In the matrix, each row denotes a character in character set C. Here $|C| = 4$ since we desire to select primers for DNA sequences. Each entry of $A_i(r,j)$, $1 \le r \le |C|$, $1 \le j \le w$, is given by

$$A_i(r,j) = \begin{cases} 1, & \text{if the position } j \text{ of } x_i \text{ is the } r\text{th character;} \\ 0, & \text{otherwise.} \end{cases}$$

The probability matrix in the Gibbs sampler method is thus modified as follows.

New probability matrix

$$Q^e_{r,j} = \sum_{i=1}^{n} A_i(r,j) \times weight(i) \tag{5}$$

The score function in the Gibbs sampler method is also changed as follows.

New score function

$$q_z = \sum_{j=1}^{w} \sum_{r=1}^{4} Q^e_{r,j} log \frac{Q^e_{r,j}}{D_r} \tag{6}$$

With the above modification, we integrate the Gibbs sampler method [10], the ACO strategy [5], and Liao's algorithm for finding motifs [11] to form our algorithm for finding local motifs as follows.

Algorithm LMF (Local Motif Finding)

Input: A set I of n DNA sequences $S_1, \cdots, S_n$, and the local motif length w.

Output: A set M of local motifs with length w, where each input sequence is covered by at least k local motifs.

Initiation: $M = \Phi$. For every sequence S_i, set *same*(i) to 0 and cover count k_i to 0, where k_i denotes the number of local motifs covering S_i.

Phase 0: Set all parameters and initialize pheromone trails.

Phase 1: Apply ACO to find the best sample (consensus substring) x^b which are constructed by a set of ants.

1. Each ant h randomly produces a sample string x^h with length w. The probability of selecting one character is calculated by Formula 2.
2. Each ant h compares sample x^h with each sequence S_i to find the best matched substring x_i^h in S_i, where $|x_i^h| = w$. Then each ant h gets a set $L^h = \{x_1^h, x_2^h, \cdots, x_n^h\}$.
3. For L^h, create a probability matrix Q^e with size $|C| \times w$.
4. Apply Formula 6 to calculate the score q^h of each substring set L^h. Update the pheromone with Formula 3.
5. Find the best score q^b among all q^h's. And update the best sample x^b accordingly. If the best sample x^b is not improved after some predefined iterations, go to Phase 2; otherwise, repeat this phase.

Phase 2: Construct a set of good candidate positions in every sequence by the best sample x^b.

By scanning each sequence S_i and comparing it to x^b, find a set U_i of $\frac{|S_i|}{\theta}$ good candidate positions, where θ is a predefined parameter. Here each candidate position represents the substring starting at that position with length w. And then let x_i be the best position in U_i.

Phase 3: Input those good candidate positions into the Gibbs sampler method to find the best substring set L^b.

1. Randomly choose one sequence in I, called S^*, whose best position is x_{S^*}. Let $T = \{x_1, x_2, \cdots, x_n\} - \{x_{S^*}\}$.
2. For T, create a probability matrix Q^e with size $|C| \times w$.
3. For each candidate position z of U_{S^*}, apply Formula 6 to calculate the score of q_z.
4. Randomly choose a starting position z in U_{S^*} with probability proportional to q_z. Update the best substring set L^b if the best score q^b is replaced.
5. If the best substring set L^b is not updated after some predefined iterations, go to Phase 4; otherwise, repeat this phase.

Phase 4: Add the local motif of L^b into M and check if M can cover each given sequence at least k times.

1. Find the local motif m^b of L^b. Add m^b into M.
2. For each S_i, find the most matched substring with m^b. Calculate the number of mismatches between the substring and m^b. Update *same*(i) and k_i accordingly.
3. If each $k_i \geq k$, terminate the algorithm; otherwise, go to Phase 0.

After Algorithm LMF terminates, we obtain the set of local motifs that cover each given sequence at least k times. In fact, the set of local motifs we find are primer candidates in the primer selection problem. Then, we can solve the MPS problem by transforming M as the input of the binary integer programming, which was proposed by Hsieh et al. [7].

4 An Example Illustration of Our Algorithm

The Gibbs sampler method [10] and Liao's algorithm [11] are designed to find the motif in the global view. In other words, the motif they found are consensus in almost all given sequences. However, our goal is to find the minimal set of local motifs which can cover the given sequences. The "local" view here means that each local motif should cover some of given sequences, not necessarily all, with very highly matched or even exactly matched.

We use the *weight* parameter to guide the motif searching in the local view. If we apply Liao's algorithm directly to find local motifs, it is a special case that the *weight* parameter is always kept a constant. But it can not work well. In the following, we use an example to illustrate our algorithm and to see the effectiveness of *weight* parameters.

Example: Illustration of Algorithm LMF.

Input: The set of sequences $I = \{S_1 =$ CGGATTTACGATG, $S_2 =$ AACTGCGATGTA, $S_3 =$ TTCGTAGCGATGA, $S_4 =$ CACCGGATACATA$\}$, and the local motif length $w = 5$.

Output: A set M of local motifs that each input sequence is covered by at least $k = 1$ local motif.

We assume that there are two substring sets that we find in the local motif searching process (Phase 3). One is $L^p = \{x_1^p = CGATG, x_2^p = CGATG, x_3^p = CGATG, x_4^p = ACATA\}$, and the other is $L^q = \{x_1^q = CGGAT, x_2^q = CTGCG, x_3^q = CGTAG, x_4^q = CGGAT\}$. In L^p, we get the local motif is $CGATG$, called l^p. Tables 1 and 2 show that the probabilistic models of L^p and L^q, respectively. For simplified demonstration, instead of Formula 6, we use an approximate calculation, which is the summation of the square of every number in the probabilistic model table. In addition, we suppose that there is no mismatch in the cover condition. For example, the approximate score of L^p is $1^2+3^2+1^2+3^2+4^2+4^2+1^2+3^2 = 62$. It is clear that l^p covers S_1, S_2 and S_3 with matches starting at positions 9, 6 and 8, respectively. In L^q, we get the local motif $CGGAT$, called l^q. The approximate score of L^q is 54. l^q covers S_1 (starting at position 1), and S_4 (starting at position 4). In this algorithm, it finds the best substring set with the highest score. In this example, it will select L^p in Phase 3 in the first iteration.

The initial *weights* are equal in all sequences. As the program executes, the *weight* of the sequence which has been covered by local motifs several times will decrease. If the sequence is covered by no or fewer local motifs ($same(i)$ is small), the *weight* of the sequence is larger than others. In the searching process of our algorithm, the trend is converging to the local motif in the sequence which has larger weight. So it gives us more chance to find the local motif to cover the sequence which has not been covered yet.

In this example, we assume that there are two iterations. In the first iteration, it finds that the score of L^p is 62 and the score of L^q is 54. So it will select L^p. After we choose L^p, it would cover sequences S_1, S_2, and S_3. So the values of $same(1)$, $same(2)$, $same(3)$, and $same(4)$ are 3, 3, 3, and 0, respectively. Thus $weight(1)$, $weight(2)$, $weight(3)$, and $weight(4)$ become 1/4, 1/4, 1/4, and 1. In the second iteration, we assume that we find L^p and L^q again. The new probabilistic models are shown in Tables 3 and 4, respectively. The approximate scores of L^p and L^q are 173/16 and 189/16, respectively. L^q will be selected in this iteration. When L^q is selected, S_1 and S_4 will be covered. In this example, we choose L^p in the first iteration and L^q in the second iteration to cover all given sequences. In summary, $M = \{l^p, l^q\}$.

Next, we apply Hsieh's method [7] to solve the MPS problem by transforming M as the input of binary integer programming (BIP). For more complete explanation, we add one more local motif into M. Assume M now becomes $\{X_1 =$ CGATG covers S_1, S_2, S_3; $X_2 =$CGGAT covers S_1, S_4; $X_3 =$ TTCGT covers S_2, $S_3\}$.

The motif coverage can be represented by the following coverage matrix

$$C = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

where each entry $C_{i,j} = 1$ if and only if sequence S_i is covered by local motif X_j. Another binary variable y_j is involved. $y_j = 1$ if X_j is selected, and $y_j = 0$ if otherwise.

The MPS problem becomes a BIP as follows.

$$Minimize \ \{y_1 + y_2 + y_3\}$$
$$\text{subject to } \Sigma_{j=1}^{3} C_{i,j} y_j \geq 1,\ 1 \leq i \leq 4,\ y_j = 0 \text{ or } 1$$

After the transformation, a BIP algorithm or software package, such as Lindo, can be invoked to solve the problem. In this example, we can choose the minimum set of X_1 and X_2 (or X_2 and X_3) to satisfy the above inequalities.

BIP can help us to get the minimum primer set. In Hsieh's method, they used the fixed sliding window size to scan each given sequence for finding possible candidates. For example,if we are given $n = 100$ sequences of length 1000 with sliding window size $w = 10$, then it will produce $((1000 - 10 + 1) \times 100) = 99100$ possible candidates (decision variables). If mismatches are allowed, there will be more decision variables. It will be difficult to solve a BIP with too many decision variables.

Before solving the BIP, we apply algorithm LMF to produce possible good primer candidates as the input of BIP. So the solution space has been reduced, and it makes the BIP computation easier.

Table 1: The probabilistic model of L^p in the first iteration.

-	Position 1	Position 2	Position 3	Position 4	Position 5
A	1	0	4	0	1
T	0	0	0	4	0
C	3	1	0	0	0
G	0	3	0	0	3

Table 2: The probabilistic model of L^q in the first iteration.

-	Position 1	Position 2	Position 3	Position 4	Position 5
A	0	0	0	3	0
T	0	1	1	0	2
C	4	0	0	1	0
G	0	3	3	0	2

Table 3: The probabilistic model of L^p in the second iteration.

-	Position 1	Position 2	Position 3	Position 4	Position 5
A	1	0	7/4	0	1
T	0	0	0	7/4	0
C	3/4	1	0	0	0
G	0	3/4	0	0	3/4

Table 4: The probabilistic model of L^q in the second iteration.

-	Position 1	Position 2	Position 3	Position 4	Position 5
A	0	0	0	3/2	0
T	0	1/4	1/4	0	5/4
C	7/4	0	0	1/4	0
G	0	3/2	3/2	0	1/2

5 Experimental Results

In this section, we test our method with two experiments. One is artificial data, and the other is real biological data. On the real domain, we take two gene family data to test our algorithm. Our hardware environment is PC with AMD Athlon XP 1800+ CPU. Software environment is based on Windows 2000 operation system with Borland C++ Builder. And we use Lindo 6.1 for windows application software in BIP. The newest version of Lindo in 2004 can accept at most 32000 constraints, 200000 variables and 20000 integer variables. To compare the performance with other methods, we also apply BIP to Hsieh's method [7] (the exhaustive search method) and Liao's algorithm [11].

5.1 Experiments on Artificial Domains

First we describe how we generate the artificial data, and then we will show the experimental results. There are six testing cases in our experiments with different numbers of input sequences and different lengths. The numbers of sequences in the testing cases are 50, 75, and 100. The lengths of sequences are 500 and 1000. We first generate a set of long random sequences, and some short random patterns of length 12 with alphabets $\{A, G, C, T\}$. Next, we insert the short patterns into the long sequences randomly as the local motifs that we want to find. In different testing case, we insert different numbers of patterns. For example, in testing cases 1 and 2 of Table 5, there are ten distinct patterns, and each sequence is inserted randomly with k patterns, where k is randomly selected between 1 and 5. In these six testing cases, at most 1 mismatch is allowed in one local motif.

In the exhaustive search method (Hsieh's method [7]), the set of possible primer patterns consists of all possible substrings of length w, where w is the length of the desired primer. In our experiments, w is set to be 12. In other words, we can scan each sequence and get the first possible primer pattern from the substring starting at position 1, the second pattern from substring starting at position 2, and so on. In our experiments, the substrings that do not appear no more twice, allowing at most one mismatch, are removed first. Thus the candidate set of the exhaustive search is not so huge. But it is still much larger than ours.

As we can see that the difference of our solution and the exhaustive search is small. In testing cases 1, 2 and 4, the solutions we get are almost equal to those obtained by the exhaustive search method. It is natural that our method cannot get better solutions than the exhaustive search method. But, our solutions are comparable to those obtained by the exhaustive search. It is also shown in the table that our execution time is much less than the exhaustive search.

We also test another artificial data with short random patterns of length 17. The experimental results are shown in Table 6. In these three testing cases, at most 2 mismatches are allowed in one local motif.

In summary, in our experiments on artificial data, our solutions are near to the optimal solutions obtained by the exhaustive search. And the time required for our algorithm is reduced drastically.

5.2 Experiments on Real Domains

In this subsection, we show experimental results on two human gene families, Tripartite Motif gene family [1] and TBX gene family [2], in Table 7. Tripartite Motif gene family has 33 sequences with lengths between 227 and 3384. TBX gene family has 17 sequences, in which 16 sequences have lengths between 545 and 2172, but the other sequence TBX-7 has not been submitted to Genebank. The length of primer, w, is assumed to be 17. We also assume that at most two mismatches are allowed in order to find more universal primers, but mismatches are not allowed on the $3'$ end of the primer. We set the rule that the first five positions in the $3'$ end of the primer have to be matched exactly.

As we can see in Table 7, for Tripartite Motif gene family, the solution that we get is only one more than that obtained by the exhaustive search method. For the other case, we even get the same result as that obtained by the exhaustive search method. The execution time required by our algorithm is reduced drastically. Since each gene family consists of similar sequences, it is easier to find the common patterns than random sequences. Thus, the difference between our solution and optimal solution is usually less than that in random sequences. We can conclude that our method is more suitable for solving multiple PCR in the gene family.

Table 5: Experimental results of artificial data. The length of the short pattern is 12, and the number of mismatch is allowed at most 1. N: The number of sequences; $NPAT$: The number of distinct patterns; K: The number of patterns inserted into every sequence; P: The average number of patterns required to cover all sequences.

	Sequences	Inserted patterns		Exhaustive search		Our method	
Testing case	N/Length	$NPAT$	K	Time(sec.)	P	Time(sec.)	P
1	50/500	10	$1 \sim 5$	179	8	26	9.13
2	50/1000	10	$1 \sim 5$	935	7	75	8.49
3	75/500	15	$1 \sim 5$	406	9	86	10.98
4	75/1000	15	$1 \sim 5$	2118	9	205	10.41
5	100/500	20	$1 \sim 5$	724	15	240	17.33
6	100/1000	20	$1 \sim 5$	3776	13	481	16.17

Table 6: Experimental results of artificial data. The length of the short patterns is 17, and the number of mismatches is allowed at most 2. N: The number of sequences; $NPAT$: The number of distinct patterns; K: The number of patterns inserted into every sequence. P: The average number of patterns required to cover all sequences.

	Sequences	Inserted patterns		Exhaustive search		Our method	
Testing case	N/Length	$NPAT$	K	Time(sec.)	P	Time(sec.)	P
1	50/1000	10	$1 \sim 5$	1049	7	130	7.36
2	75/1000	15	$1 \sim 5$	2374	11	496	13.45
3	100/1000	20	$1 \sim 5$	4237	17	663	18.47

Table 7: Experimental results of two gene families. The length of the primer is 17, and the number of mismatches is allowed at most 2. The first five positions in the $3'$ end of the primer have to be matched exactly. N: The number of sequences; T: Execution time (second); P: The average number of patterns required to cover all sequences.

		Gene family	
		Tripartite Motif	TBX
N		33	16
Lengths		$227 \sim 3384$	$545 \sim 2172$
Exhaustive search	T	1835	210
	P	10	3
Our method	T	462	53
	P	11	3

6 Conclusion

To obtain the good primer candidate set, we design a fast algorithm to find the set of local motifs. Our algorithm integrate the Gibbs sampler method, the ACO strategy and Liao's algorithm with the helpful *weight* parameters, which can guide us to get local motifs in the local view. Hsieh et al. [7] proposed an exhaustive search method for solving the MPS problem by transforming the problem into a binary integer programming. However, their approach produces too many decision variables. With our method, the size of the set of the primer candidates (decision variables) can be reduced drastically. Our experimental results show that our solutions are near to the optimal solutions obtained by the exhaustive search. And the time required for our algorithm is reduced drastically. Our algorithm is more suitable for solving multiple PCR in the gene family, since the sequences in a gene family usually share many common patterns.

In this paper, we do not consider the chemical properties of primers. In order to use our algorithm in the realistic applications, we can add more constraints into our algorithm, such as complementary sequences, melting temperatures, and GC-content. We can scan all patterns to estimate their biological constraints on primers. Then the suitable set with some constraints is generated, and we can apply our method to this set. Thus Our algorithm is also suitable for the realistic applications.

References

[1] http://www.informatics.jax.org/mgihome /nomen/genefamilies/trim.shtml.

[2] http://www.gene.ucl.ac.uk/nomenclature /genefamily/TBX.shtml.

[3] T. Bailey and C. Elkan, "Unsupervised learning of multiple motif in biopolymers using expectation maximization," *Machine Learning*, Vol. 21, pp. 51–80, 1995.

[4] K. Doi and H. Imai, "Greedy algorithm for finding a small set of primers satisfying cover length resolution condition in PCR experiments," *Proc. of the 8th Workshop on Genome Informatics*, Tokyo, Japan, pp. 43–52, 1997.

[5] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53–66, 1997.

[6] F. Hillier and G. Lieberman, *Introduction to Operations Research.* McGraw-Hill, Seven ed., 2001.

[7] M. H. Hsieh, W. C. Hsu, S. K. Chiu, and C. M. Tzeng, "An efficient algorithm for minimal primer set selection," *Bioinformatics*, Vol. 19, No. 2, pp. 285–286, 2003.

[8] Y. J. Hu, S. B. Sandmeyer, and D. F. Kibler, "Detecting motifs from sequences," *In Proceedings of the 16th International Conference on Machine Learning (ICML)*, Bled, Slovenia, pp. 181–190, 1999.

[9] T. kampke, M. Kieninger, and M. Mecklenburg, "Efficient primer design algorithms," *Bioinformatics*, Vol. 17, No. 3, pp. 214–225, 2001.

[10] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment," *Science*, Vol. 262, pp. 208–214, Oct. 1993.

[11] Y. J. Liao, C. B. Yang, and S. H. Shiau, "Motif finding in biological sequences," *in Proc. of 2003 Symposium on Digital Life and Internet Technologies*, Tainan, Taiwan, pp. 18–27, Sep. 2003.

[12] C. Linhart and R. Shamir, "The degenerate primer design problem," *Bioinformatics*, Vol. 18, pp. s172–s180, 2002.

[13] P. Nicodeme and J. Steyaert, "Selection optimal oligonucleotide primers for multiplex PCR," *Proc. Fifth International Conference on Intelligent Systems for Molecular Biology*, Halkidiki, Greece, pp. 210–213, 1997.

[14] W. pearson, G. Robins, D. Wredgs, and T. Zhang, "A new approach to primer selection in polymerase chain reaction experiments," *Proc. International Conference on Intelligent Systems for Molecular Biology*, Cambridge, England, pp. 285–291, 1995.

[15] W. pearson, G. Robins, D. Wredgs, and T. Zhang, “On the primer selection problem in polymerase chain reaction experiments,” *Discrete Applied Mathematics*, Vol. 71, pp. 231–246, 1996.

[16] E. Rocke and M. Tompa, “An algorithm for finding novel gapped motifs in DNA sequences,” *Proc. of the Second Annual International Conference on Computational Molecular Biology*, New York, USA, pp. 228–233, Mar. 1998.

[17] G. Stormo and G. Hartzell, “Identifying protein-bindign sites from unaligned DNA fragments,” *National Academy of Sciences*, Vol. 86, pp. 1183–1187, 1989.

[18] A. Talaat, P. Hunter, and S. Johnston, “Genome-directed primers for selective labeling of bacterial transcripts for DNA microarray analysis,” *Nature Biotech*, Vol. 18, pp. 679–682, 2000.

[19] A. Tobin and J. Dusheck, *Asking about Life.* Harcourt College Publishers, Second ed., 2001.