Planted *(l, d)* - Motif Finding using Particle Swarm Optimization

U.Srinivasulu Reddy
Department of Computer Applications
National Institute of Technology
Tiruchirappalli, Tamil Nadu, India

Michael Arock
Department of Computer Applications
National Institute of Technology
Tiruchirappalli, Tamil Nadu, India

A.V.Reddy
Department of Computer Applications
National Institute of Technology
Tiruchirappalli, Tamil Nadu, India

ABSTRACT
In Bioinformatics, Motif Finding is one of the most popular problems, which has many applications. Generally, it is to locate recurring patterns in the sequence of nucleotides or amino acids. As we can't expect the pattern to be exact matching copies owing to biological mutations, the motif finding turns to be an NP-complete problem. By approximating the same in different aspects, scientists have provided many solutions in the literature. The most of the algorithms suffer with local optima. Particle swarm optimization (PSO) is a new global optimization technique which has wide applications. It finds the global best solution by simply adjusting the trajectory of each individual towards its own best location and towards the best particle of the swarm at each generation. We have adopted the features of the PSO to solve the Planted Motif Finding Problem and have designed a sequential algorithm. We have performed experiments with simulated data it outperforms MbGA and PbGA. The PMbPSO also applied for real biological data sets and observe that the algorithm is also able to detect known TFBS accurately when there are no mutations.

General Terms: Evolutionary Optimization Techniques, Bioinformatics, Computational Biology.

Keywords: Motif Finding, Particle Swarm Optimization (PSO), Swarm Intelligence (SI), Transcriptional Factor Binding Sites (TFBS), Planted Motifs.

1. INTRODUCTION
A gene is a segment of DNA that is the blueprint for protein. Basically, the control of gene regulation is determined by the chemical reactions which are, in turn, controlled by the shape and electrostatic charges of the molecules involved. Unfortunately, this information was not available. In 1950, Francois Jacob and Jacques Monod first discovered regulation genes, in Paris. These genes provide the instructions for creating proteins to control the expression of the other structural genes and play a key role in gene expression.

In order to regulate the gene expression process, a molecule called transcription factor will bind to a short substring in the promoter region of the gene. We call this substring as a binding site of the transcription factor. A single transcription factor can be bound to multiple binding sites. We refer to these binding sites as "Motifs". Motifs are fundamental functional elements in proteins. These patterns are vital for understanding gene function, human disease, and may serve as therapeutic drug target. Motifs can be used to determine evolutionary and functional relationships of the genes. Motifs vary in lengths, positions, redundancy, orientation and bases. Finding these short sequences (motifs or signals) is a fundamental problem in molecular biology and computer science with important applications such as knowledge-based drug design, forensic DNA analysis, and agricultural biotechnology [1].

Motif Finding is the process of locating the meaningful patterns in the sequence of DNA, RNA or Proteins. The patterns are not exact copies due to biological reasons. So, the motif finding problem turns to be an NP-Complete Problem. It is one of the key areas of interest for a number of researchers. There are different types of motifs in the literature namely: Sequential Motifs, Gapped motifs, Structured Motifs, Planted Motifs and Network Motifs. A number of methods, algorithms and tools have been developed in the recent years to solve these problems. Gibbs Sampler [2] and MEME [3] are most widely used in practice to solve the motif finding problem and these methods are local search methods. For planted motifs, Random Projections [4] and Pattern Branching [5] got better results compared to others. The complete survey of DNA motif finding algorithms, methods and different approaches are presented in [6]. All these methods suffered from the problem of local optima. In the recent days many Evolutionary Computational Techniques/Evolutionary Algorithms (EAs) are being tried with different coding schemes and different objective functions to eliminate local optima. Among these, Genetic Algorithm (GA) is one of the widely used algorithms to find motifs. Though GA's help overcome the problem of local optima, it is only to some extent and it is possible only at the cost of exercising more operators [7].

The Swarm Intelligence (SI) is a recent emerging technology to solve optimization problems. It has a lot of scope to handle complex problems in the field of Bioinformatics and Computational Biology [8]. The Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are the two popular techniques in SI. The characteristics of PSO promise to arrive at a new optimization framework which probes in the neighborhood regions. In addition, it is done systematically to explore the neighborhood profiles.

In recent years, PSO has been used to solve different types of motif finding problems. The following are a few to mention with their advantages and disadvantages.

Hardin and Rouchka proposed a hybrid motif discovery approach framed combining Particle Swarm Optimization (PSO) and the Expectation Maximization (EM) algorithm. They used PSO to generate a seed for the EM algorithm [9]. This method still suffered with local optima.

Zhou et al. formulate the Transcriptional Factor Binding Sites (TFBS) as a Combinatorial Optimization Problem. Then, they applied a hybrid PSO (HPSO) to solve the challenging issue in upstream regions of genes regulated by Octamer binding sites. They developed two local search operators and one recombination

mutation operator in HPSO [10]. These results bring out some putative binding sites motifs, but not all the binding sites. The same author and his team demonstrated how to used evolutionary computing method to discover the binding sites. Then, they proposed a novel algorithm IPSO-GA by integrating an improved PSO with GA to search sequence motifs from co-expressed genes regulated by the NF-Kb transcription factor [11]. Their experimental results help find putative binding sites, but not to discover the true motifs.

X. Chang et al. proposed a novel framework to use EA to identify transcriptional factor binding sites. They introduced two EA techniques GA and PSO and also presented two different coding methods to solve this problem [12]. These methods can find correct binding site motifs than Gibbs Sampling and MEME. In their second experiment, both GA and PSO fail to predict the binding site of the sequence 17.

B. Chang et al. applied PSO algorithm to protein sequence motif discovery problem [13]. The results were compared with PROSITE database and they obtained global optimum protein sequence motifs but these were not more meaningful in biological sense.

In this paper, we adopt the features of the PSO to solve the Planted Motif Finding Problem. We perform experiments with simulated *(l, d)*-planted motif challenging instances of (10, 2), (11, 2), (12, 3), (15, 4), (16, 5), (18, 6), (20, 7), (30, 11) and (40, 15) and real biological data sets and observe that the algorithm works better for the longer motif instances. Our approach is also able to detect known TFBS accurately. The rest of the paper is organized as follows: Section 2 imparts a background with preliminaries of motif finding problem. Section 3 describes PSO background and its applicability to Motif Finding Problem. Section 4 discusses the PSO algorithm for planted motif problem Section 5 deals with experimental results and Section 6 presents the conclusion with future work.

2. BACKGROUND

Before defining the Motif Finding Problem, let us consider the definitions of string and substring. A string S is an ordered list of characters written contiguously from left to right. For any string S, S[*i..j*] is the (contiguous) substring of S that starts at position *i* and ends at position *j* of S. A motif is a substring s of length *l* that may or may not present in a given string S. An occurrence of a (*l, d*)-motif is a substring s of length *l* that varies at most *d* positions from the motif.

Motif Finding in general, is locating recurring patterns in the sequence of nucleotides or amino acids. Formal definition of motif finding problem is as follows: Let S = {s_1, s_2 …s_T} be a sample of *T N*-letter biosequences, each sequence containing an (*l, d*)-motif, i.e., a motif of length *l* with *d* mismatches (mutations). In this paper, we consider the planted motif finding problem which is preciously defined by Pevzner and Sze [14, 15]. In this, each input string S_i contains a planted occurrence of an (*l, d*)-motif, having an initial positions $j \in \{1, 2...N-l+1\}$, where *N* is the length of the string S_i. Here, we assume that all strings are of the same length. The aim of the planted motif problem is to find all the occurrences of the (*l, d*)-motifs that appear in each of the T input strings without knowing, *a priori*, the motif.

We will now formally define the Planted Motif Finding Problem as the following input/output requirements.

Input: A set of T strings {s_1, s_2 …. s_T} each of length *N* over alphabet {a, c, g, t}, where each string contains a planted occurrence of the (*l, d*)-motif.

Output: A set of P best starting positions {p_1, p_2 …. p_T } where the planted (*l, d*)-motif occurs in T strings.

All the solutions to motif finding problem essentially follow a three-step method. The steps are: Representing the sequences, determining suitable objective function(s), and employing appropriate search strategies. There are two common ways of representing motifs: Consensus sequence and Position Weight Matrix (PWM) or Probabilistic Matrix representations. Whilst first method takes symbols by majority, the second method assigns probability of each nucleotide occurring at each position of the motif sequence. Basically, there are two major classifications of methods for motif finding: Scanning for known motifs and Employing Statistical or Combinatorial methods. First method aims at searching in a database (already formed) which is a collection of known transcription factors as well as their DNA binding sites and profiles. For example, TRANSFAC database and PROSITE are two important databases among many. This method is not suitable, when we need to find new sites. So, we employ either statistical or combinatorial approach. The literature says the combinatorial approach is a better approach than the statistical approach. Gibbs Sampler, MEME, AlignACE, BioProspector , CONSENSUS , and TEIRESIAS are a few to mention. These approaches are unable to solve the Challenge Problem proposed by Pevzner and Sze in 2000. They defined the challenge problem as follows: Given a sample of *n=20* sequences, each *N=600* nucleotides long with an implanted motif of length *l =15* with *d=4* mutations, find the motif. Since then, various approaches have solved the motif challenge problem. The WINNOWER, SP-STAR [15], MERMAID [16], ROJECTIONS, MULTIPROFILER, Pattern Branching and Profile Branching are the most frequently referred among them. All the above said algorithms share a common protocol which is a step-by-step procedure of planted motif discovery shown in figure 1. Sequences with annotated motifs, how these motifs hide when affected with pathogen, how to find these hidden motifs by motif search programs and compare the predicated motifs with annotated motifs for the accuracy, are the steps.

2.1 Preliminaries

For better understanding of planted motif finding problem, we recall some of the definitions here:

2.1.1 Objective Function

Computational methods are defined based on the objective functions we choose. The purpose of an objective function is to approximate the correlation between sequence patterns and their biological meaning in terms of mathematical function. The objective functions are only heuristics. After objective function is determined, the goal is to find the patterns of high objective function value. To reach this goal, two important associated issues are employed: pattern representation and search strategy. We use *score* as the objective function in this problem.

$$\textbf{Score} = \sum_{j=1}^{l} M_{P(s)}(j)$$

Where P(s) is profile matrix corresponding to starting positions S. $M_{P(s)}(j)$ is the largest count in column *j* of P(s).

2.1.2 Hamming Distance

It is the number of changes applied to a sequence to obtain another sequence. For example, If V = ATTGTC and W=ACTCTC, then d (V, W) = 2.

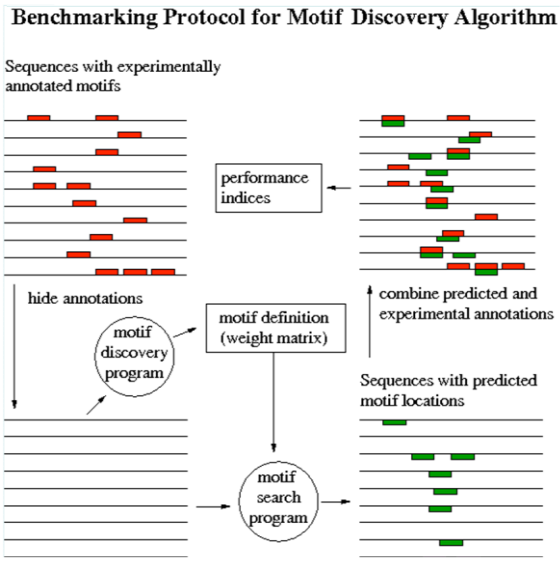

Fig.1. Benchmarking protocol for Planted Motif Finding.

3. BACKGROUND ON PARATICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) method is an evolutionary optimization technique first developed by Kennedy and Eberhart [17] in 1995. It finds the global best solution by simply adjusting the trajectory of each individual towards its own best location and towards the best particle of the swarm at each generation.

Particle Swarm Optimization is one of the optimization techniques under EAs. The particle swarm imitates a kind of social optimization. Given a problem, a suitable evaluation method, called fitness function is to be formulated and also a communication structure or social network that lets individuals to interact among them. Then, with the help of random inferences as start points, we trigger an iterative process. It aims at improving the candidate solutions step-by-step by finding the best fatnesses' and remembering their locations. These are essentially local best successes. Now, with the interaction among neighbors, escorted by these successes we can move toward the globally best success.

The position and velocity of the i^{th} particle in the n-dimensional search space can be represented as $X_i = (x_{i1}, x_{i2}, x_{i3}\ldots x_{in})$ and Velocity $V_i = (v_{i1}, vi2, vi3\ldots v_{in})$ respectively. Each particle has its best position $P_i = (p_{i1}, p_{i2}, p_{i3}\ldots p_{in})$ corresponding to the personal best fitness value obtained so far at time t, with reference to the user defined objective function. The global best particle, which represents the fittest particle found so far at time t in the entire swarm, is denoted by p_g. The new velocity of the each particle is calculated according to the following equation.

$$V_{in}(t+1)= \omega \cdot v_{in}(t)+c_1 \cdot rand_1() \cdot (p_{in}-x_{in})+c_2 \cdot rand_2() \cdot (p_g-x_{in}) \quad (1)$$

Where c_1 and c_2 are acceleration coefficient constants, ω is called inertia factor and $rand_1$ ( ) and $rand_2$ ( ) are two sparely generated uniformly distributed random numbers in the range [0, 1].

At each iteration (generation), the position of each particle is updated according to the following equation.

$$X_{in}(t+1) = x_{in}(t) + V_{in}(t+1) \quad (2)$$

3.1 The pseudo-code of the procedure is as follows:

For each particle
    Initialize particle
End
do
  For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness value (pBest) in history set current value as the new pBest
  End
  Choose the particle with the best fitness value of all the particles as the gBest
  For each particle
    Calculate particle velocity according equation (1)
    Update particle position according equation (2)
  End
While maximum iterations or minimum error criteria is not attained

3.2 Flow chart

The following flowchart shows the detail flow of the PSO algorithm. First, it generates the initial population and then evolves local best particles, i.e., pBest's. If the termination condition occur or no change in the results, then it selects the global best particle i.e., gBest and prints the results. Otherwise, it calculates new velocity and new positions for particles by using above formulae (1) and (2).

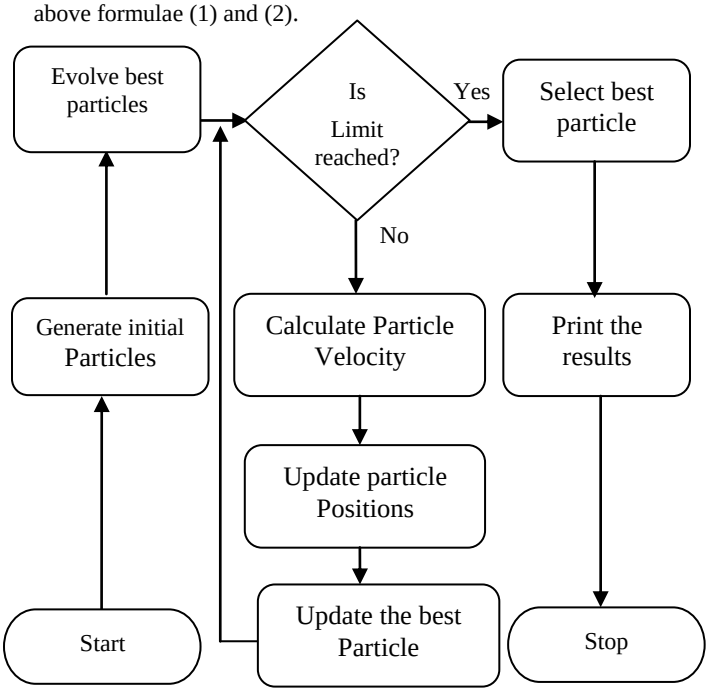

Fig.2.The basic structure of PSO

4. PARTICLE SWARM OPTIMIZATION FOR PLANTED *(l, d)* - MOTIF FINDING

1. [Initialization]

   NUM_ITERATION and POP_SIZE can be altered by user

   POP_SIZE is recommender to be above 30

   Initial velocity of the each individual is 0

2. Generate a random number from each sequence in the range 1 to *N-l+1* called position of the particle x_i in the n-dimensional search space.

   $c_1, c_2 \leftarrow 2$

   $\omega \leftarrow 0.6$

3. for i $\leftarrow$ 1 to NUM_ITERATION

   do

   rand1( ) and rand2( ) are the random numbers in the range [0,1]

   for i $\leftarrow$ 1 to POP_SIZE

   do

   Ten offsprings are obtained from moving the particle *i* 5 positions right and 5 positions left. Find the score for particle *i* and its children. $pBest_i$ is the best scored position of the particle *i* and its children

   End for

   gBest is the best scored position from all $pbest_i$'s

$$V_{in}(t+1) \leftarrow \omega \cdot v_{in}(t)+c_1 \cdot rand_1() \cdot (p_{in} - x_{in})+c_2 \cdot rand_2() \cdot (p_g - x_{in})$$

$$X_{in}(t+1) \leftarrow x_{in}(t) + V_{in}(t+1)$$

If the particle's positions go beyond the extreme, follow wrap-around. Using new particles continue the next iteration until no change in particles or maximum number of iterations.

i $\leftarrow$ i + 1

End for

In the above pseudo-code, we set an initial population by selecting a random starting position from each sequence in the range [1, *N-l+1*] and assign it to a particle. Repeat the same process for whole population. After getting the initial population, we generate ten offsprings by moving a particle five positions left and five positions right. We evaluate the fitness function for the parent and its offsprings by applying the *score* as an objective function. The positions of sequences with the maximum *score* value are called pBest's. we repeat the same process for all the particles. After getting all pBests, find the maximum *score* among all the pBests, the maximum is the gBest. Once we get the pBest and gBest values, we update the velocity and position by substituting these values in the formulae (1) and (2). If the particle's positions go beyond the extreme, follow wrap-around. We continue the above process for the new particles until we get the planted motif or we reach the maximum number of iterations. The main difficulty we face in the algorithm implementation is to set an appropriate parameter value for PSO algorithm, i.e., Vmax value. Based on the objective function the results also vary. We should give due importance to select a suitable objective function. By trial and error, we have selected parameters for proposed algorithm as follows:

Table 1 Parameters of proposed algorithm (MbPSO)

Parameter description	Parameter Values
Size of Swarm : POP_SIZE	30
Self-recognition coefficient: c1	2
Social coefficient: c2	2
Inertia Weight: w	0.6
Maximum Velocity: Vmax	5

5. EXPERIMENTAL RESULTS

5.1 Experimental Set Up:

To test our proposed algorithm with simulated data, we generate *(l, d)*-planted motif challenging instances of (10, 2), (11, 2), (12, 3), (15, 4), (16, 5), (18, 6), (20, 7), (30, 11) and (40, 15) as follows: first, a motif length *l* is generated by choosing *l* at random. Second, we construct *N=20* background sequences each of length *T=600*. Third, we mutate the motif *l* by randomly choosing *d* position. Finally we implant the mutated motif at randomly generated motif occurrences in each sequence. We used Intel core 2 Duo processor at 2.66GHZ with 2GB of RAM and 80 GB hard disc for implementing the algorithm.

An experimental comparison of two encoding schemes for planted motif finding problem by using GA is presented in [7]. They are position-based GA, PbGA and Motif-based GA, MbGA. Their results show a clear solution quality improvement of the motif-based representation over the position-based representation. MbGA and PbGA did not use to find real biological data and they took more time to find implanted motifs.

In this paper, we compare the results of these two encoding schemes with our proposed PSO-based algorithm for Planted Motif Finding called PMbPSO, because all the three algorithms belong to the same family of the EAs. Table 2 shows the average computation time ($\overline{T}$) in seconds of MbGA, PbGA and PMbPSO. The PMbPSO algorithm is run 30 times for each instance and the average of time taken by PMbPSO shows that PMbPSO outperforms MbGA and PbGA. The MbGA and PbGA are unable to find longer motifs after (18, 6) onwards. The proposed algorithm PMbPSO results clearly show that it is able to find longer motifs.

Table 2 The average computation time ($\overline{T}$) of MbGA, PbGA and PMbPSO

(l, d)	MbGA	PbGA	PMbPSO
(10,2)	81.67	11.1	0.98
(11,2)	91.50	11.77	1.12

(12,3)	131.50	12.4	1.18
(15,4)	155.17	31.1	1.23
(16,5)	335.10	12.39	1.25
(18,6)	-	-	1.36
(20,7)	-	-	1.38
(30,11)	-	-	1.85
(40,15)	-	-	2.23

Table 3 shows the standard deviation (S. D.) values for the MbGA, PbGA and PMbPSO. PMbPSO works better for longer motifs. It takes less S.D. values when we increase the length of the motifs and the number of mismatches, whereas MbGA, PbGA did not show any results for longer size motifs in their results.

Table 3 The standard deviation for MbGA, PbGA and PMbPSO

(l, d)	S.D. MbGA	S.D. PbGA	S.D. PMbPSO
(10,2)	17.95	2.58	1.00
(11,2)	21.57	3.05	1.33
(12,3)	28.92	2.34	1.38
(15,4)	55.55	10.66	1.28
(16,5)	95.52	2.35	0.94
(18,6)	-	-	1.12
(20,7)	-	-	1.06
(30,11)	-	-	0.56
(40,15)	-	-	0.72

The following figure 3 shows the average time taken by the different *(l, d)*-planted motifs. The graph clearly depicts how time varies with respect to the size of the motif length and the number of mutations. Time is directly proportional to motif lengths and its mutations. When the size of *(l, d)* increases, time also increases.

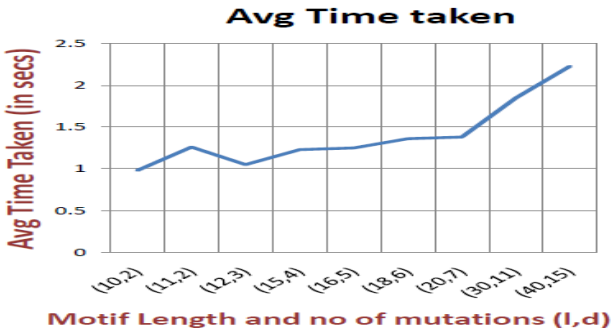

Fig. 3 Average Time taken by different *(l, d)*-planted motifs.

The following figure 4 shows the average number of mismatches taken by the different *(l, d)*-planted motifs. The graph clearly indicates that the proposed algorithm works better for the longer size motifs. The proposed algorithm is able to find longer size motifs with minimum number of mismatches.

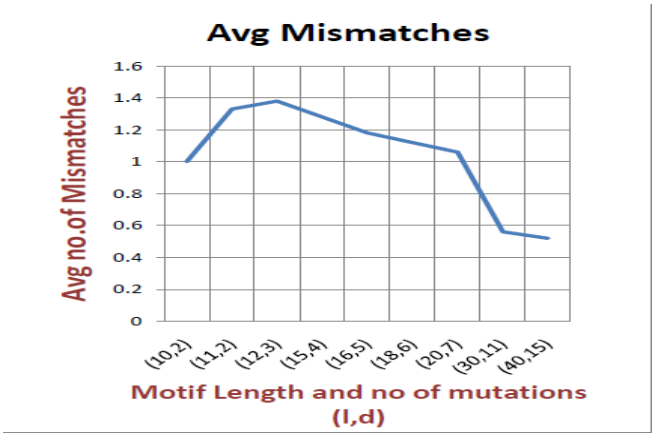

Fig. 4 Average Mismatches taken by different *(l, d)*-planted motifs.

We have also tested the proposed algorithm (PMbPSO) for real biological sequences stored in the public database SCPD [18]. The sequence lengths *T*, the motifs length *l* were the same as those of the published in SCPD. We have tried values for d, 0 and 1. Experimental results are shown in Table 4. The PMbPSO could find the motifs for these data sets within one second for each data set. The PMbPSO is able to find motifs when *d=0* exactly.

Table 4 Experimental results on real biological data sets.

Transcription Factor binding sites	**Published Motifs**	**Motifs Discovered by PMbPSO**	
		d=0	***d=1***
GCR1	CWTCC	CTTCC	CTTCC
GATA	CTTATC	CTTATC	CTTATC
CCBF,SCB,SW16	CNCGAAA	CACGAAA	CACGAAA
CuRE, MAC1	TTTGCTC	TTTGCTC	AAGCAAA
GCFAR	CCCGGG	CCCGGG	CCCAGG

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have adopted the features of the PSO to solve the Planted Motif Finding Problem and have designed a sequential algorithm. We have performed experiments with simulated *(l, d)*-planted motif challenging instances of (10, 2), (11, 2), (12, 3), (15, 4), (16, 5), (18, 6), (20, 7), (30, 11) and (40, 15). Our proposed algorithm outperforms MbGA and PbGA with respect to the average time taken and S.D. values. The results also show that the proposed algorithm works better for longer size motifs. The PMbPSO also applied for real biological data sets and observe that the algorithm is also able to detect known TFBS accurately when there are no mutations.

In future, we wish to extend the same for more number of instances and also for the hard instances like (9, 2) (11, 3) (13, 4) (15, 5) and (17, 6). We face a difficulty in the algorithm implementation in setting an appropriate parameter value for PSO algorithm, i.e., Vmax value. We should give due importance to select a suitable objective function, because the outcomes are

more dependent on the objective function. Hence, we plan to employ multi-objective functions to solve different types of motif finding problems.

7. REFERENCES

[1] Xiong, J. 2006. Essentials of Bioinformatics, Cambridge press.

[2] Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A., and Wootton, J. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignments, Science, 262, 208-214.

[3] Bailey, T., and Elkan, C. 1995. Unsupervised learning of multiple motifs in biopolymers using expectation maximization, Mach. Learning, 21, 51–80).

[4] Bubhler, J., and Tompa, M. 2001, Finding motifs using random projections, Proceedings of the Fifth Annual International Conference on Research in Computational Molecular Biology, (69-76).

[5] Price, A., Ramabhadran, S., and Pevzner, P. 2003. Finding subtle motifs by branching from sample strings, Bioinformatics, (149-155).

[6] Modan, K., Das and Dai, H. 2007. A survey of DNA motif finding algorithms, BMC Bioinformatics, (1-13).

[7] Mart´ınez-Arellano, G., and. Brizuela, C.A. 2007. Comparison of Simple Encoding Schemes in GA's for the Motif Finding Problem: Preliminary Results, Springer-Verlag Berlin Heidelberg, (22–33).

[8] Hassanien, A., Mariofanna, G., Milanova, Tomasz G., Smolinski, and Abraham, A. 2008. Computational Intelligence in Solving Bioinformatics Problems: Reviews, Perspectives, and Challenges, (1-48).

[9] Hardin, C. T., and Rouchka, E. C. 2005. DNA Motif Detection Using Particle Swarm Optimization and Expectation-Maximization, proc. IEEE, (181- 184).

[10] Zhou, W., Zhou, C., Liu, G., and Huang, Y. 2005. Identification of Transcription Factor Binding Sites Using Hybrid Particle Swarm Optimization, Springer-Verlag Berlin Heidelberg, (438–445).

[11] Zhou, W., Zhu, H., Liu, G., Huang, Y., Wang, Y., Han, D., and Zhou, C. 2005. A Novel Computational Based Method for Discovery of Sequence Motifs from Coexpressed Genes, International Journal of Information Technology, 11, 8, (75-83).

[12] Chang, X., Zhou, C., Li, Y., and Hu, P. 2006. Identification of Transcription Factor Binding Sites Using GA and PSO, Proc.6th Int. Conf on Intelligent Systems Design and Applications, (1-5).

[13] Chang, B. 2004. Particle Swarm Optimization for Protein Motif Discovery, in Genetic Programming and Evolvable Machines, 5, (203-214).

[14] Gusfield, D. 1997. Algorithms on Strings, Trees and Sequences, Computer Science and Computational Biology. Cambridge University Press, Cambridge.

[15] Pevzner, P., and Sze, S.-H. 2000. Combinatorial approaches to finding subtle signals in DNA Sequences. Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology, (269–278).

[16] Hu,Y. 2003. Finding subtle motifs with variable gaps in unaligned DNA sequences, Computer Methods and Programs in Biomedicine, 70, (11–20).

[17] Kennedy, J., and Eberhart, R. C. 1995. Particle Swarm Optimization, Proc. of the IEEE International Conference on Neural Networks (1942-1948).

[18] Zhu, J., and Zhang M.Q. 1999. A promoter database of the yeast Saccharomyces cerevisiae. Bioinformatics, 15, (607-611).