A PARTICLE SWARM OPTIMIZATION ALGORITHM FOR

FINDING DNA SEQUENCE MOTIFS

APPROVED BY SUPERVISING COMMITTEE:

_____

Jianhua Ruan, Ph.D., Chair

_____

Weining Zhang, Ph.D.

_____

Jeffery von Ronne, Ph.D.

Accepted: _____

Dean, Graduate School

DEDICATION

This thesis is dedicated to my dear beautiful wife Ruting and my dear selfless parents; thank you for providing me with constant inspiration.

A PARTICLE SWARM OPTIMIZATION ALGORITHM FOR FINDING DNA SEQUENCE MOTIFS

by

CHENGWEI LEI, B.S.

THESIS
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In partial Fulfillment
Of the Requirements
For the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Department of Computer Science
August 2008

ACKNOWLEDGEMENTS

Particular thanks to my advisor, Ruan, Jianhua, who advised and gave me lots of research ideas and directions. Under his supervision I learn how to do a research and how to make the research better.

I am very grateful to the Dr. Zhang, Weining, Dr. Ruan, Jianhua and Dr. Ronne, Jeffery von for being on my advisory committee and reviewing my thesis.

Special thanks are also due to Department Administrative Assistant Erika Martínez De Rizo, Department Senior Administrative Associate Cindy Murphy, and Jennifer Bigler.

Needless to say, I have missed many of you who have helped me before. Please accept my sincere thanks to all of you.

August 2008

A PARTICLE SWARM OPTIMIZATION ALGORITHM FOR

FINDING DNA SEQUENCE MOTIFS

Chengwei Lei, B.S.
The University of Texas at San Antonio, 2008

Supervising Professor: Jianhua Ruan, Ph.D.

Discovering short DNA motifs from a set of co-regulated genes is an important step towards deciphering the complex gene regulatory networks and understanding gene functions. Despite significant improvement in the last decade, it still remains one of the most challenging problems in both computer science and molecular biology. Most of the computational approaches for finding motifs belong to one of two broad categories: stochastic optimization algorithms based on position specific weight matrices, or combinatorial search algorithms based on consensus patterns. Recent evaluation studies have shown that methods based on position specific weight matrices tend to stuck in local optima, while combinatorial search algorithms are typically limited to small data sets and short motifs only. In this work, we propose a novel algorithm based on a population-based stochastic optimization technique called Particle Swarm Optimization. Compared to previous methods, our algorithm represent motifs by consensus patterns, thus avoiding many of the pitfalls associated with weight matrix-based methods. On the other hand, our algorithm does not require exhaustive enumeration of all consensus sequences, yet can still quickly converge to an optimal or near optimal solution. Preliminary results on both simulated and real biological data sets are encouraging, showing that our method is both more efficient and more accurate than several existing algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

One of the most important mechanisms to regulate gene functions is on the transcription level, where the expression of genes is mediated by the binding of transcription factors (TF) to the promoter sequences of genes. Identifying common transcription factor binding sites (TFBS) from a set of putatively co-regulated genes is an important step towards deciphering the complex gene regulatory networks and understanding the tissue/condition-specific functions of genes.

Although TFBS can be detected with an array of wet-lab experiments, such as DNase footprinting and Chromatin immunoprecipitation., such experiments are often expensive, time-consuming, and do not scale well to the whole genome. In practice, many studies have relied on a combination of computational and experimental approaches, thanks to many large-scale genome sequencing projects and gene expression profiling efforts. In the first step, a set of genes that are believed to be co-regulated are obtained computationally or experimentally (e.g. based on gene expression data or Chromatin immunoprecipitation assays). Given the promoters of these putatively co-regulated genes, one then apply computational motif finding algorithms to identify short DNA sequences ("motifs") that are statistically over-represented in these promoters. However, accurate identification of these motifs is difficult because they are typically short (8-15 bps) compared to the promoter sequences (usually several hundred to thousands of bases long). Furthermore, there is often a great variability among the binding sites of any given TF, and the biological nature of the variability is not yet well understood. Despite these difficulties, in the past decade a large variety of computational methods have been developed, many of which have been proven useful in predicting true binding sites.

The existing motif finding algorithms often differ from one another in their ways of defining motifs, the objective functions for calculating motif significance, and the search

techniques used to find the optimal (or near optimal) motifs. Many of these algorithms can be classified into one of two broad categories: stochastic searching algorithms based on position specific weight matrix (PSWM) motif representations, and combinatorial search algorithms based on consensus sequence motif representations. Examples of the first category include the well-known programs such as MEME, AlignACE , GibbsSampler , BioProspector, while the second category can be exemplified by Weeder, YMF, MultiProfiler, and Projection. For surveys of the existing methods and assessments of their relative performance, see *Assessing computational tools for the discovery of transcription factor binding sites*. As can be expected, no single currently existing method stands out as the sole best in all cases. In fact, assessing the performances of these algorithms is a daunting task itself, and experiments have shown that the overall performance of motif finding algorithms is still quite low, (although this should not be misinterpreted as a discouragement for further improvement either). Nevertheless, there seems to be a slight advantage by combinatorial approaches. To test the capacity of various motif finding algorithms, Pevzner and Sze designed a set of challenging cases, the socalled *(l,d)*-motif, a set of DNA *l*-mers each of which differ from a common consensus sequence by exactly d mismatches. The motifs were then embedded into some random DNA sequences and submitted to various motif finding algorithms. It has been shown that many stochastic searching algorithms fail to recover the embedded motifs even for biologically realistic choices of parameters (e.g. a set of (15, 4)-motifs embedded in 20 sequences each with 600 bases). On the other hand, although combinatorial search algorithms have generally been shown to perform better in these challenging test cases, they typically resort to exhaustive enumeration of all or a large number of variants of consensus sequences, and are therefore limited to small data sets and short motifs only.

In this study, we propose a novel motif finding algorithm based on a population-based stochastic optimization technique called Particle Swarm Optimization (PSO). Compared to previous methods, our algorithm uses consensus as motif representation, thus avoiding many of the pitfalls associated with PSWM-based methods. On the other hand, PSO is purely stochastic, does not require exhaustive enumeration of all consensus sequences, yet can still quickly converge to an optimal or almost optimal solution. Our preliminary results on both simulated and real biological data sets have shown that our method is both more efficient and more accurate than a number of existing algorithms.

The remaining sections are organized as follows. In Chapter 2, we introduce the basic ideas of PSO and the generic PSO algorithm for those who are not familiar with this technique. In Chapter 3, we discuss the improvement that we made to the generic PSO algorithm in order to accommodate the unique nature of motif finding problems, and other algorithmic issues that we addressed. We present our experimental results in Chapter 4, and conclude in Chapter 5 with some possible future improvement.

CHAPTER 2: PARTICLE SWARM OPTIMAIZTION

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique for problem solving that is inspired by the social behaviors of organisms such as bird flocking and fish schooling. The system is initialized with a population of random solutions and searches for the optimal solution by updating iteratively. Although PSO shares many similarities with evolutionary computation techniques such as genetic algorithms, PSO differs from other evolutionary algorithms significantly on how the solutions were updated. In PSO, each potential solution, called particle, is represented by a point in the multiple-dimensional solution space. When searching for the optimum solution, particles fly around the solution space with a certain velocity (speed and direction). During flight, each particle adjusts its position and velocity according to its own experience and the experience of its neighbors. Specifically, each particle keeps track of the best solution (the position and the fitness value) it has encountered so far. This solution is called *pbest*, which stands for personal best. Each particle also keeps track of the best solution by any particle in its neighborhood. This solution is called *lbest*, which stands for local best. Many types of neighborhood structures can be implemented by emulating real social networks. In the simplest case, all the particles are directly connected to each other. Then lbest is simply the global optimum of all the particles, hence is also called *gbest*. Therefore, while each individual particle is performing a local search, the particles also communicate with other particles and learn from them, balancing exploration and exploitation.

Formally, let vectors x_i and v_i be the current position and velocity of the *i*-th particle ($0 < i < n+1$), $\dot{x}_i$ be the recorded position of pbest of the *i*-th particle, and $\dot{g}$ be the position of *gbest*. The fundamental concept of PSO consists of changing the velocity (v_i) of each particle at each

time step toward its *pbest* and *gbest* locations. Acceleration is weighted by a random number, with separate random numbers being generated for acceleration toward *pbest* and *gbest* locations. The particles update their positions and velocities based on the two equations below.

$$V_i = \omega \bullet V_i + c_1 \bullet rand_1() * (\dot{x}_i - x_i) + c_2 \bullet rand()_2 * (\dot{g} - x_i)$$
$$x_i = x_i + V_i$$

where ω is a parameter called the inertial weight, $rand_1()$ and $rand_2()$ are vectors of random numbers, usually uniformly distributed within 0 and 1. The operator * denotes entry-wise vector multiplication. It is critical to note that different $rand_1()$ and $rand_2()$ are generated at each iteration and for each particle. c_1 and c_2 are positive constants, called the acceleration constant. c_1 is a factor determining how much the particle is influenced by its *pbest*, and c_2 is a factor determining how much the particle is influenced by *gbest*.

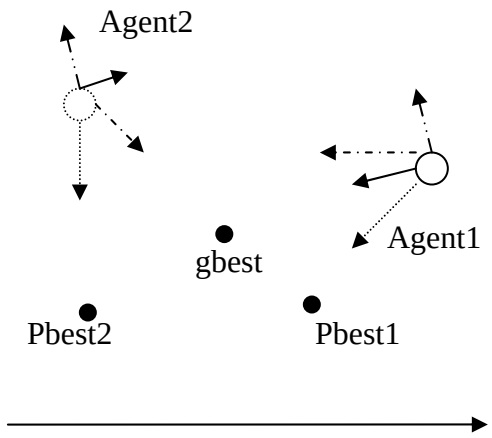

Figure 1. How PSO make a movement.

The above figure shows how personal best and global best affected a particle's next step velocity. It shows two particles above, *agent1* and *agent2*. Each one has its own personal best fitness value, *pbest1* and *pbest2* . And each agent knows the global best fitness value *gbest*. Every next step velocity's direction and value is depending on by what proportion is each agent attracted by $pbest_i$ between *gbest*. Every agent has its current velocity v_i and is pulled by $pbest_i$ and *gbest*. The vector sum of all these three will give the next step's direction and velocity

amount. When a particle is close enough to the global optimization, final *gbest*, it should have slow down so that it don't miss the *gbest* and don't go farther away. On the other hand, if a particle is far from the objective *gbest*, it should move rapidly to find out the solution in a short time.

CHAPTER 3: THE PSO-MOTIF ALGORITHM

Solution representation and fitness function

In order to apply PSO to the motif finding problem, we need to first define the solution representation (and accordingly, the search space) and the fitness function. There are several ways to represent a motif. In PSWM-based algorithms, motifs are represented by weight matrices, so the solution can be unambiguously represented by a $3l$ matrix, where l is the length of the motif.

Our focus of this paper is to develop a consensus-based algorithm, due to its relative advantage that has been shown recently. To represent a motif in consensus-based algorithms, we have two choices. First, we may represent a motif by a consensus sequence, which is a discrete l-dimensional vector. The search space consists of all the 4^l l-mers. Alternatively, a motif can be most precisely defined by the list of motif instances contained in the input sequences. Therefore, with the motif length fixed, a motif can be represented by a vector of the starting positions of the motif instances within each input sequence. Between the two representations, we opt for the second representation in this work. The main advantage of the latter is speed in updating, although both have their own advantages and disadvantages, and in some cases are equivalent. A more detailed discussion of this issue is beyond the scope of this paper.

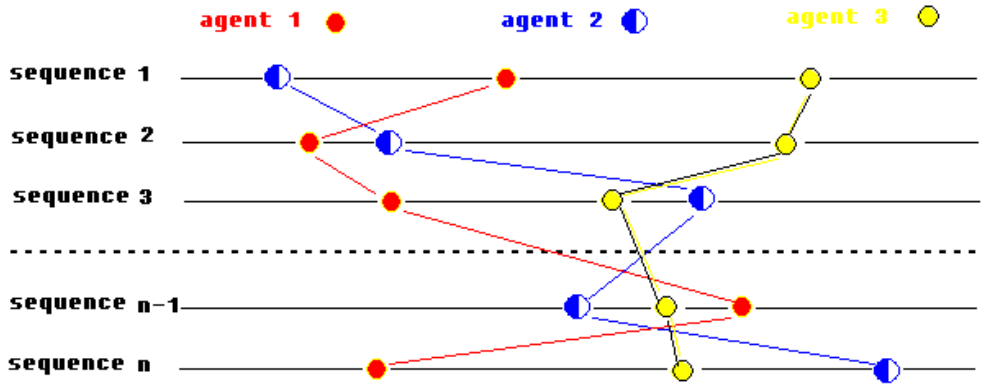

Figure 2. Relationship between each agent

To evaluate the quality of a motif, we first derive a consensus for the motif by taking the most frequent base at each position, and then measure the total number of mismatches between the individual instances and the consensus. When this fitness function is used, it is assumed that the background sequences have uniform base frequencies, which is usually not true. However, it is straightforward to design a more elaborated fitness function to take into consideration background base frequencies, with slightly increased running time.

Mismatch table

The main difficulty associated with applying the PSO algorithm to motif finding problem is that the fitness function is not smooth (in a loose sense). In a typical PSO algorithm, one wishes to control the velocity so that at the beginning stage the particles can fly around quickly inside the search space, and when a particle approaches the optimal solution, it should slow down so it can converge quickly. One can achieve this if the fitness function is continuous, since the velocity is updated according to the distances between the current position and the positions of pbest and gbest. In motif finding, the distance between two solutions has no indication of the difference of their fitness values. For example, as shown in the figure below, position 1 and position 2 are separated by 6 mismatces, while position 1 and position N have only 1 mismatch.

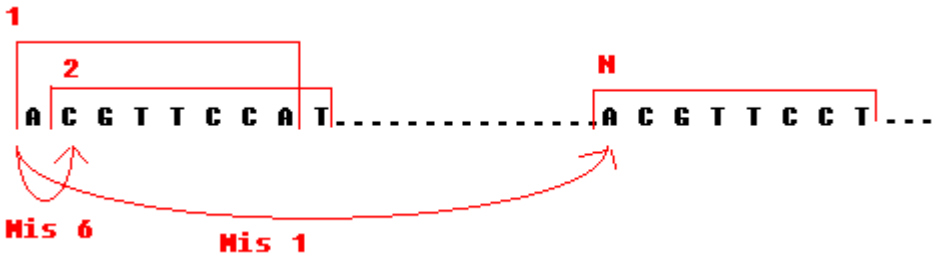

Figure 3. Relationship between motif positionsand number of mismatches

To solve this problem, we remap the neighborhood information in the solution space by constructing a dissimilarity graph of all *l*-mers in each sequence. For example, given an input sequence CTCTGCTG and motif length = 3, we can build the mismatch table as in Table 1.

Table 1. An example mismatch table

	P1	P2	P3	P4	P5	P6
P1	\	3	1	2	3	1
P2	3	\	3	2	1	3
P3	1	3	\	3	3	0
P4	2	2	3	\	3	3
P5	3	1	3	3	\	3
P6	1	3	0	3	3	\

The row index is the current position of a candidate motif in this sequence, the column index is the potential new position of the motif, and the values in the cell are the number of mismatches between the two *l*-mers starting at the positions represented by the row index and column index, respectively. For example, if the motif start position is updated from 1 to 6, the distance is only 1, meaning CTC and CTG have 1mismatch. With this table; we can reformat the sequence into a new order, which is more meaningful and useful. In the above table, the ``neighborhoods'' order of P2 is (P5), (P4), (P1;P3;P6). This table can be pre-computed and stored in the main RAM. As one table is needed for each sequence, the total space needed is $O(nL^2)$, where n is the number of sequences and L is the length of each sequence. For longer

sequences, the tables can be made sparse by keeping only the table entries smaller than a certain threshold, for example, *2d*, where *d* is the maximum number of errors expected between the motif consensus and its instances.

Update policy

In order to decide what new positions should be considered as the motif starting position, we define the velocity of a particle as the range of the allowed number of mismatches between the new and current motif instances. Let D (x_i, x_j) be the vector of numbers of mismatches between the motif instances in x_i and x_j, we use the following rule to update the velocity of a particle:

$$v_i^u \leftarrow \omega\, v_i^u + c_1^u\ \ r_1 * D(x_i, \dot{x}_i) + c_2^u\ \ r_2 * D(x_i, g)$$

$$v_i^l \leftarrow \omega\, v_i^l + c_1^l\ \ r_1 * D(x_i, \dot{x}_i) + c_2^l\ \ r_2 * D(x_i, g)$$

Given the upper and lower bounds of the velocities, v_i^u and v_i^l, we update x_i as follows. Let $x_i(j)$ be the starting position of the motif on the *j*-th sequence, and $x_i'(j)$ be the new starting position on the same sequence. In order to obtain $x_i'(j)$, we randomly pick a position from sequence *j* such that the number of mismatches between the motifs started at these two positions are within the upper and lower bound of the velocity, i.e.,

$$v_i^l(j) <= D(\ x_i(j)\ ,\ x_i'(j)\ ) <= v_i^u(j).$$

Occasional full scan and shift check

Inside the algorithm we always keep track a consensus sequence derived from *gbest* using the most frequently base at each position, and compute the fitness value of gbest. When the fitness value of *gbest* reaches a certain threshold, we use the consensus sequence to do a full scan on all sequences to check if there is an *l*-mer in the input sequence that matches the consensus sequence better than the *l*-mer in *gbest*.

Second, similar to many motif finding algorithms, the output of PSO algorithm may have a shift issue: the start positions may be one or two positions away from the real positions, and it is difficult for the algorithm to escape from such local optima. To circumvent this problem, we periodically check whether shifting the motif positions by a small number can improve the quality of the solution.

Table 2. An example shift table

Position	1	2	3	4	5	6
	A	A	C	T	C	C
	T	A	C	T	A	G
	C	A	G	T	A	T
	G	A	C	T	A	A
Before check	----------					
After check		----------				

Escape from local optima

If gbest remains stable for a large number of iterations, we consider that the algorithm has reached a local optimum or the global optimum, so we reset the algorithm. There are two ways for reset. The first strategy is to simply discard all the information so far and start the algorithm from scratch. In some cases with this strategy the algorithm might converge to the same local optimum repeatedly. Here we suggest the second strategy, which we call a "reset move". When a local optimum is detected, we move all the current solution, *pbest* and *gbest* by a random distance. We have found that this strategy can more effectively help the algorithm escape from local optima.

Termination

Like many stochastic algorithms, in many cases it is difficult to determine when the algorithm should terminate and report a solution. For simulated test cases, we typically know the total number of mismatches, so the program can stop when it reaches the threshold. In practice,

however, a user typically does not have prior knowledge about the number of mismatches. If the number is given too high, the algorithm may give up the search before it finds the optimal solution. On the other hand, if the number is given too low, the algorithm may spend most of its time trying to improve over an easily detectable global optimum. Here we implemented three methods to determine when the algorithm should terminate: value-based, time-based, and repeat-based. The value-based method is the easiest: when the solution reaches a pre-defined threshold value, the algorithm stops running and returns the solution. The time-based method terminates the algorithm after a certain amount of time has passed. Finally, we recommend the repeat-based method for real test cases. With this method, before resetting the algorithm, we compare the current gbest fitness value with the best gbest fitness value achieved during the course of the algorithm. If the current gbest is better than the history gbest, we update the history gbest; if the two fitness values and the associated motif consensuses are exactly the same, we stop the algorithm and output this result. This method is based on the intuition that, since there are typically many local optima, the probability of repeating one local optimum without an update is very low. This method is very effective when tested on the real test cases.

Post-processing

By default, the quality of a motif is evaluated by the total number of mismatches between the consensus sequence and its instances. There are a number of limitations in this basic strategy when applied to real biological sequences. First, the mismatch based quality function does not consider the background frequency, which is usually not uniform. Second and more importantly, in real TF binding motifs the mutation rates are often not uniformly distributed across every position, and not every site has the same significance in determining the binding strength. For example, many TF binding sites consist of two short conserved regions, separated by a small gap.

Therefore, a mismatch in the gapped region should be penalized much less than a mismatch in the conserved region. However, in the consensus based method, we are forced to select a representative for each position, and the number of mismatches is computed based on that representative.

To address these problems, we apply a post-processing procedure to improve the final motif returned by the PSO algorithm. Given the *gbest* returned by the algorithm, we construct a position-specific weight matrix. This matrix is then used to scan all input sequences. This scan will likely update some of the motif instances. We then recompute the position specific weight matrix and repeat the scan, until the solution does not vary. With this method, the matching/mismatching score is weighted by the information content contained in each position, and therefore the more conserved positions will have more contribution to the selection of the binding sites. Furthermore, we can also take into account the background base frequencies when scoring a binding site against the weight matrix.

Flow chart

Figure 4 shows the overall structure of the PSO-motif algorithm. The individual steps have been described in the previous subsections.

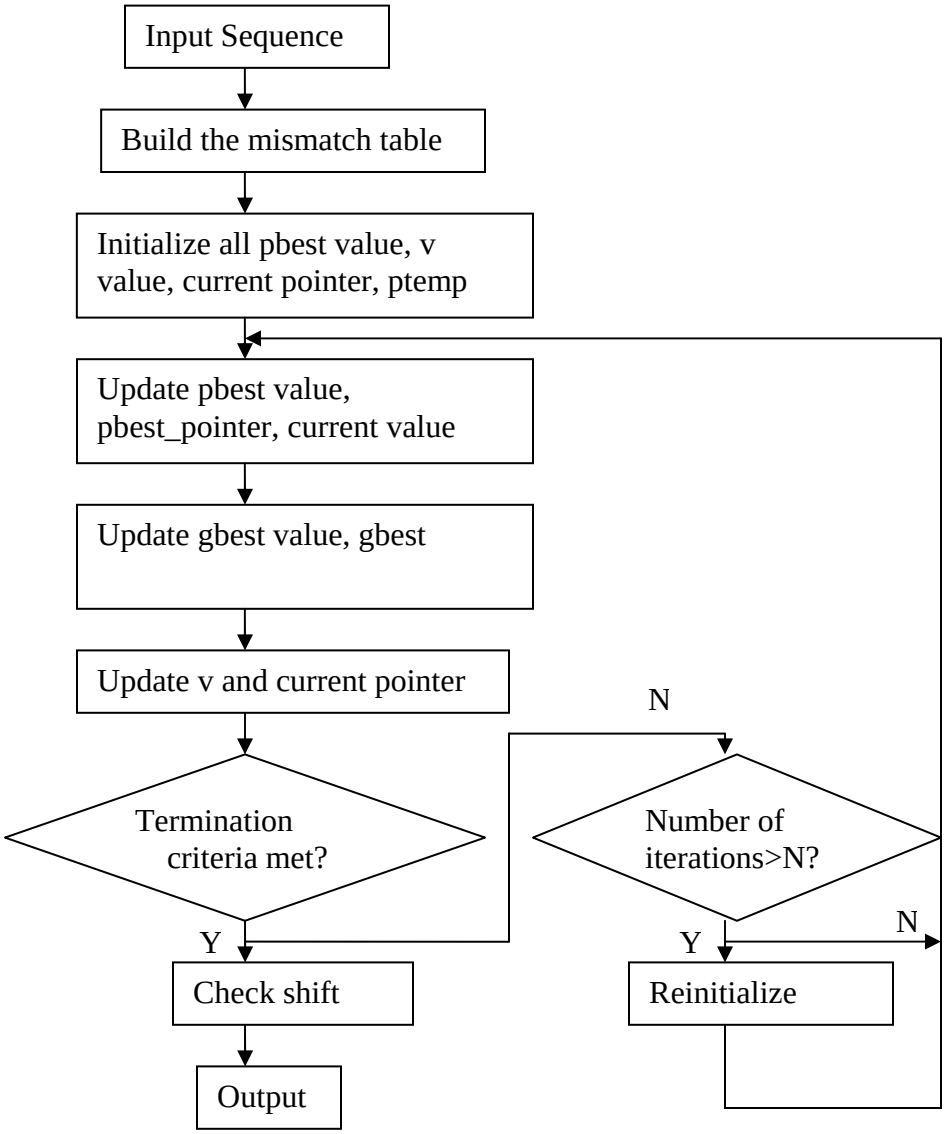

Figure 4. Flow chart of PSO- motif algorithm

CHAPTER 4: EXPERIMENTAL RESULTS

We have implemented our algorithm in C. To evaluate the performance of our algorithm, we tested it on two types of DNA sequences. The first type of test data consists of simulated data sequences, also known as the *(l, d)*-motif challenging problem. The second type of data contain real promoter sequences from E. coli and human genes containing known TF binding sites that have been determined experimentally.

Simulated data sets

To objectively compare with the existing algorithms, we first tested our algorithm on a set of simulated data sets. We synthesized problem instances as follows. First, a motif consensus of length *l* was generated by randomly picking *l* bases. Second, we randomly selected *d* positions (without replacement) from the consensus and mutated the base at each position to a random base, which could be the same as the original base. This generates one instance of the motif. We repeated this process to obtain *t* instances. Third, we randomly generated *t* background sequences of length n each. Finally, we assign each motif instance to a random position in a background sequence. This procedure generates t sequences, each containing exactly one instance of the motif. All random choices were made independently and with equal base frequencies.

We first focused on the (15, 4)-motifs challenging problem, which is one of the most tested problem instances by many programs. We fixed the number of sequences to 20, and varied the length of each sequence from 400 to 1000. Since our algorithm is stochastic, we repeated our algorithm multiple times with random restart, and terminated it once it first found the motif. Table 3 shows the time needed by our algorithm to solve these instances. The results are the average of 20 independent runs on different data sets. Table 3 also shows the running time of two of the best combinatorial-search motif finding algorithms: Projection, a random projection based

algorithm, and Weeder, a suffix tree based enumeration algorithm. The running time of Weeder was taken from the original paper which was published almost 7 years ago. According to the original authors, the running time was based on an 89%success probability. Unfortunately we were not able to repeat the experiments ourselves since the program is not available. The program Projection was downloaded from the original author's website (http://cse.wustl.edu/jbuhler) and we tested the program on the same data set as our algorithm. PSO-motif and Projection were able to recover all the embedded motifs with 100% accuracy. Weeder is generally slower than Projection and PSO-motif. While PSO-motif and Projection have similar running time, the running time of Projection increases more rapidly with the length of the input sequences. Furthermore, the number of iterations of random projections in Projection has to be predetermined, which has to be estimated from the motif length and the number of mismatches. If a user does not know the number of mismatches in advance, one either has to use a large number of iterations which requires a prolonged running time, or use a small number of iterations with the risk to miss the real motif.

Table 3. Running time of PSO, Weeder, and Projection on (15,4)-motif challenge problems

Sequence length	400	500	600	800	1000
Weeder	<1m	125s	200s	450s	15m
Projection	9s	23s	42s	162s	418s
PSO	20s	20s	35s	65s	530s

Next, we compared PSO, Projection and MotifEnumerator, a pattern-driven motif enumeration algorithm, on a series of challenge problems with varying motif lengths and number of errors. The program MotifEnumerator was downloaded from the original author's website (http://faculty.cs.tamu.edu/shsze/motifenumerator/). Again both PSO-motif and Projection were

able to recover the embedded motifs with 100% accuracy. The test cases fixed the number of sequences to 20, and the length to 600. As shown in Table 4, although Projection is more efficient than PSO for shorter motifs, its advantage vanishes when motif length increases to about 15, and it is slower than PSO for motifs longer than 17. The running time and space requirement of MotifEnumerator are exponential to *l*. Therefore, for small motif lengths (e.g., *l* = 11), MotifEnumerator solves the problem very efficiently; however, when *l* increases to 15, the program aborted with an out of memory exception on our testing computer with 2GB RAM.

Table 4. Running time of PSO and Projection on other motif challenge problems

(l,d)	(11,2)	(13,3)	(15,5)	(17,5)	(19,6)
Projection	4s	13s	42s	94s	174s
PSO	25s	34s	62s	73s	107s
MotifEnumerator	5s	119s	-	-	-

Real biological sequences with known motifs

Finally, we tested our algorithm on several biological samples with known TF binding motifs.

(1) The first sample is the binding sites for the cyclic AMP receptor protein (CRP), which functions as a transcription factor in Escherichia coli. The data set contains 18 sequences, each 105 bp long, which contain 23 sites that have been experimentally determined. This dataset has been previously analyzed by Stormo and Hartzell, Lawrence and Reilly and Liu.

(2) The second test sample is the binding site for the estrogen receptor (ER), which is a ligand-activated enhancer protein that binds to specific DNA sequences called estrogen response elements (EREs) with high affinity and activates gene expression in response to estradiol. The

data set includes 25 genomic sequences, each of which is 200 bp long and contains a single known ERE.

(3) The final data set includes 25 mammalian sequences of 200 bp long that are known to contain binding sites for the transcription factors in the E2F family.

It is important to note the following difference between them and the simulated data set. (1) The background base frequencies are not uniform in these real data sets; (2) The number of mutations are not known; and (3) The mutation rates vary at different positions. Therefore, it is interesting to test whether any of these characteristics in real sequences may pose additional difficulty for our consensus-based motif finding algorithm.

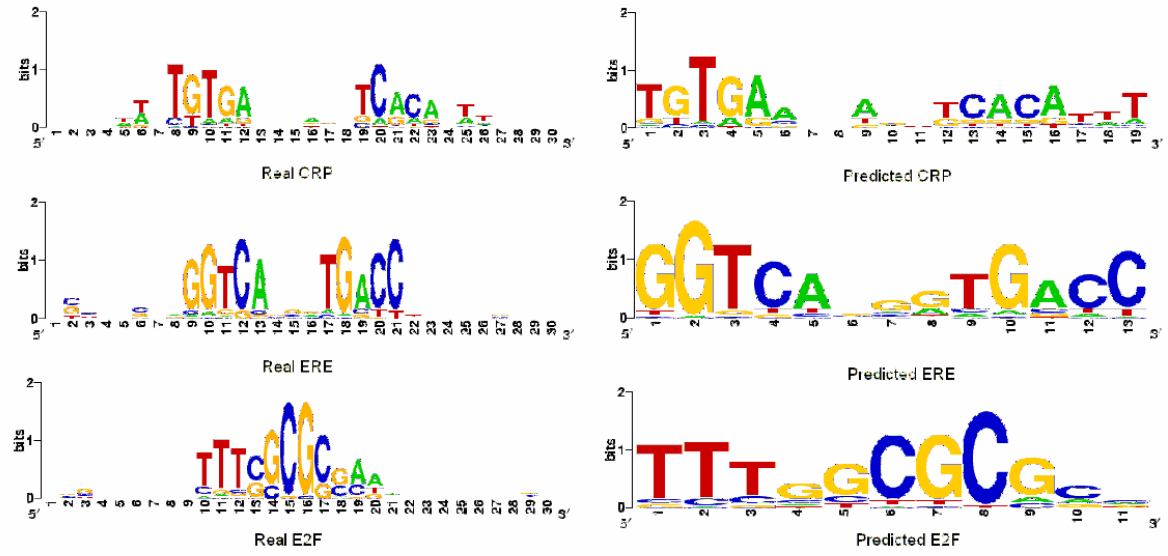

Figure 5. Comparison between known motifs and predicted motifs.

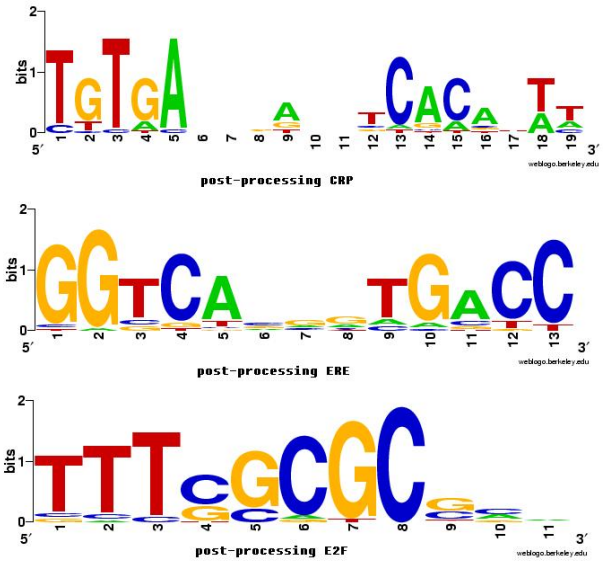

Figure 6. Motif after the post-processing

Figure 5 and Figure 6 show the sequence logos of the known motifs together with the flanking regions, compared with the motifs predicted by our algorithm. As shown, our algorithm has recovered the consensuses of all three real motifs. On the other hand, the probabilities of observing a given base at some position are different between the predicted and the real motifs, especially for the CRP motif. Further investigation shows that the reason for this partial discrepancy is often due to (1) each input sequence may contain more than one motif, while our algorithm only allowed one occurrence per sequence; (2) our algorithm does not take into consideration background base frequencies. In the CRP data set, the 18 sequences contained a total of 24 binding sites. Furthermore, the sequences contain a very high AT content (61%).

Our algorithm included several binding sites that match to the consensus sequence equally well as or better than the true sites, but have higher AT contents. Therefore, although the binding sites identified by our algorithm have a smaller number of mismatches to the consensus than the real binding sites, the former may be less biologically interesting. Therefore, we applied to post-processing procedure described in Chapter 3 G. Figure 6 shows that the post-processing did not

change the consensus sequence, but fine-tuned the motif instances and the predicted motifs with post-processing are more similar to the real ones than those without post-processing are.

CONCLUSIONS AND DISCUSSION

In this work, we have proposed a novel algorithm for finding DNA motifs based on a population-based stochastic optimization technique called Particle Swarm Optimization (PSO). Compared to previous methods, our algorithm represent motifs by consensus patterns, thus avoiding many of the pitfalls associated with weight matrix-based methods. On the other hand, our algorithm does not require exhaustive enumeration of all consensus sequences, yet can still quickly converge to an optimal or near optimal solution. Preliminary results on both simulated and real biological data sets are encouraging, showing that our method is both very efficient and accurate. When applied to simulated challenge problems, PSO is slower than two existing algorithms in easy cases, while much faster in more difficult cases involving longer input sequence or longer motifs and more mutations.

Furthermore, our algorithm does not require the number of mismatches to be given as a parameter, which is more useful in practice. For real biological sequences, our method with post-processing has successfully identified the known motifs and most of the binding sites. Our studies have shown that PSO is a reliable and efficient technique for solving the difficult motif-finding problem, and we are looking into applying it to other challenging problems in computational biology.

REFERENCES

[1] T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol,* 2:28–36, 1994.

[2] B. Berman, Y. Nibu, B. Pfeiffer, P. Tomancak, S. Celniker, M. Levine, G. Rubin, and M. Eisen. Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the drosophila genome. *Proc. Natl Acad. Sci. USA,* 99:757–762, 2002.

[3] J. Buhler and M. Tompa. Finding motifs using random projections. *J Comput Biol*, 9:225–42, 2002.

[4] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on antennas and propagation.*, 6:58–73, 2002.

[5] G. Crooks, G. Hon, J. Chandonia, and S. Brenner. Weblogo: a sequence logo generator. *Genome Res*, 14:1188–90, 2004.

[6] R. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Proc. 2001 Congr. Evolutionary Computation*, 2001.

[7] R. Eberhart, Y. Shi, and J. Kennedy. Swarm Intelligence. Morgan Kaufmann, 2001.

[8] M. Frith, U. Hansen, J. Spouge, and Z.Weng. Finding functional sequence elements by multiple local alignment . Nucleic Acids Res., 32:189–200, 2004.

[9] J. Hu, B. Li, and D. Kihara. Limitations and potentials of current motif discovery algorithms. Nucleic Acids Res., 33(15):4899–913, 2005.

[10] U. Keich and P. Pevzner. Finding motifs in the twilight zone. Bioinformatics, 18:1374–81, 2002.

[11] A. Kel, O. Kel-Margoulis, P. Farnham, S. Bartley, E. Wingender, and M. Zhang. Computer-assisted identification of cell cycle-related genes: new targets for e2f transcription factors. J. Mol. Biol., 309:99–120, 2001.

[12] J. Kennedy and R. Eberhart. Particle swarm optimization. In Proc. IEEE Conf. Neural Networks IV, 1995.

[13] C. Klinge. Estrogen receptor interaction with estrogen response elements. Nucleic Acids Res., 29:2905–2919, 2001.

[14] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. Science, 262:208–14, 1993.

[15] C. Lawrence and A. Reilly. An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins., 7:41–51, 1990.

[16] N. Li and M. Tompa. Analysis of computational approaches for motif discovery. Algorithms Mol Biol., 1:8, 2006.

[17] J. Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. J. Am. Stat. Assoc., 94:958–966, 1994.

[18] X. Liu, D. Brutlag, and J. Liu. Bioprospector: discovering conserved dna motifs in upstream regulatory regions of co-expressed genes. Pac Symp Biocomput., pages 127–38, 2001.

[19] G. Pavesi, G.Mauri, and G. Pesole. An algorithm for finding signals of unknown length in dna sequences.. Bioinformatics, 17:S207–14, 2001.

[20] S. S. Pevzner, PA. Combinatorial approaches to finding subtle signals in dna sequences. Proc Int Conf Intell Syst Mol Biol., 8:269–78, 2000.

[21] J. Robinson, S. Sinton, and Y. Rahmat-Samii. Particle swarm optimization in electromagnetics. IEEE Transactions on antennas and propagation., 52:397–407, 2004.

[22] F. Roth, J. Hughes, P. Estep, and G. Church. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mrna quantitation. Nat Biotechnol., 16:939–45, 1998.

[23] S. Sinha and M. Tompa. Discovery of novel transcription factor binding sites by statistical overrepresentation. Nucleic Acids Res., 30:5549–60, 2002.

[24] G. Stormo and G. Hartzell. Identifying protein-binding sites from unaligned dna fragments. Proc. Natl Acad. Sci., 86:1183–1187, 1989.

[25] S.-H. Sze and X. Zhao. Improved pattern-driven algorithms for motif finding in dna sequences. In Proc. of the 2005 Joint RECOMB Satellite Workshops on Systems Biology and Regulatory Genomics, Lecture Notes in Bioinformatics, volume 4023, pages 198–211, 2006.

[26] M. Tompa, N. Li, T. Bailey, G. Church, B. De Moor, E. Eskin, A. Favorov, M. Frith, Y. Fu, W. Kent, V. Makeev, A. Mironov, W. Noble, G. Pavesi, G. Pesole, M. Rgnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. Nat Biotechnol, 23:137–44, 2005.

VITA

Chengwei Lei became a Master student in Computer Science Department in University of Texas in San Antonio since Aug 2006. Prior to coming to San Antonio, he is an Engineer in SinoRail Information Engineering Group, from June 2005 to July 2006. In 2005 June, he received his Bachelor degree in Computer Science and Technology; and also received a Bachelor degree in Applied Mathematics, from Beijing University of Aeronautics and Astronautics.