

SOME IMPLEMENTATION ISSUES OF HEURISTIC METHODS FOR MOTIF EXTRACTION FROM DNA SEQUENCES

Liang Yang, Enli Huang and Vladimir B Bajic

Knowledge Extraction Lab, Institute for Infocomm Research, 21 Heng Mui Keng Terrace,
Singapore 119613

yangliang75@hotmail.com, stuhe@i2r.a-star.edu.sg, bajicv@i2r.a-star.edu.sg

ABSTRACT

Frequently, a set of conserved (common) motifs in a group of functionally related biological sequences (DNA/RNA or proteins) has a specific biological function. Determination of compact groups of these common patterns is a prerequisite for their efficient modeling. We discuss some of the most crucial aspects of computer implementation of three heuristic algorithms which can be used in computational extraction of such conserved motifs from a set of unaligned DNA/RNA sequences. The algorithms included are tabu search, simulated annealing and a population-based genetic algorithm. A server with these algorithms implemented is available as a public web application free for academic and non-profit users at http://sdmc.i2r.a-star.edu.sg/DRAGON/Motif_Search/. This tool can be directly applied in determination of functional patterns in DNA and RNA.

Keywords: *ab initio* DNA motif discovery, heuristic methods, genetic algorithms, simulated annealing, tabu search.

1. INTRODUCTION

One of the most common problems in bioinformatics is alignment of biological sequences, either DNA/RNA or proteins. This is done for the reasons of finding parts of different biological sequences which share certain level of similarity, since such information is in a broad use in determining sequence functionality. Many efficient programs for local and global alignments are developed such as BLAST (Altschul *et al.*, 1990), FASTA (Pearson and Lipman, 1988, Pearson, 1990), ClustalW (Higgins *et al.*, 1994), etc. However, for the purpose of modeling particular groups of patterns which are common to a set of sequences, with the aim of using these models to search for the new members of the hypothetically functional group, these general alignment programs do not provide the most useful solutions, mainly due to the facts that they are based on far too many constraints imposed by the speed of program and due to the form in which they express the similarity between the sequences. Moreover, they are not very efficient in aligning short motifs, such as those usually representing transcriptional regulatory elements.

Many biological sequences which belong to a group of functionally related genes or proteins, usually contain a number of biologically active sequence patterns shared among some and sometimes all members of the functional group. A typical example represents promoters of a group of co-expressed genes which contain many common transcriptional regulatory elements which also share similar positional organization such as order and mutual distances (Werner, 1999). Determination of the best models of these common patterns is a great challenge.

There are several surveys (Brazma *et al.*, 1998; Rigoutsos *et al.*, 2000; Brejova *et al.*, 2000) related to motif extraction problems. Several computer programs exist which can be used for this purpose such as MEME (Baily and Elkan, 1994), GibbsDNA (Lawrence *et al.*, 1993), CONSENSUS (Hetz and Stormo, 1999), AlignACE (Roth *et al.*, 1998), TEIRESIAS (Rigoutsos and Floratos, 1998), SAM (Hughey and Krogh, 1998), SPLASH (Califano, 2000), Weeder Web (Pavesi *et al.*, 2004) and probabilistic suffix trees (Berejano and Yona, 2001). Interesting to note, these programs produce mutually quite different results. This is not necessarily a bad thing as this may be useful for the users to make selections to suite

their need most appropriately. While it is not clear yet how short functional DNA motifs, such as transcription factor binding sites, have evolved in complex organisms, evolutionary algorithms with their heuristic nature provide relatively simple paradigms that could be close to some possible ways of evolution of such short functional motifs. While we do not imply that short functional DNA motifs have actually evolved in manners similar to heuristic algorithms, we do want to exploit such ideas and use them for determination of motif groups, which may prove efficient in some cases. Some heuristic methods such as genetic algorithms (GAs) (Denning, 1992, Goldberg, 1989; Holland, 1975; Reeves, 1993, 1997, Mitchell, 1996) operate by using population generation models and thus are very close to the way in which some biologically active DNA motifs could evolve. It is thus interesting to analyze the use of such algorithms in pattern selection problems as they fit well into discrete optimization which makes a natural framework for computational sequence pattern selection. We will discuss a/ GA, b/ tabu search (TS) (Glover, 1986; 1989, 1990; Glover and Laguna, 1997), and c/ simulated annealing (SA) method (Egglese, 1990; Fleischer, 1995; Kirkpatrick *et al*, 1983; Johnson *et al*, 1989; Tovey, 1988). The drawback of these algorithms is lower speed, but consistency of the extracted pattern groups is usually considerably higher than what is obtained with expectation maximization (EM) or with Gibbs sampling. We discuss some of the most important aspects of technical implementation of these methods. These methods can be directly applied in determination of functional patterns in DNA/RNA. They are implemented in a system named Dragon Motif Search (DMS), which is available as a public web application free for academic and non-profit users at http://sdmc.i2r.a-star.edu.sg/DRAGON/Motif_Search/.

2. HEURISTIC ALGORITHMS IN APPLICATION TO DNA MOTIF EXTRACTION

Heuristics represent an essential ingredient of methods in applied optimization. Heuristics could be the only applicable approach to many complex problems encountered in practice. The reason to use heuristics is to ensure that a reasonable, near-optimal solution to the optimization problem in question can be found in an acceptable time. Heuristics are equally applicable to both constrained and unconstrained optimization problems. In principle, there are two types of heuristics each differing in the fundamental approach used. Standard heuristics attempts to improve a single current solution in some way. Contrary to this, the other approach, population heuristics, operates with population of solutions and combine them mimicking genetic reproduction process to generate new solutions. We will discuss implementation issues for our problem of two standard heuristics, SA and TS, as well as a population heuristic based on GA. All three methods produce very good results.

2.1 Data and goal

Our data set represents a collection of DNA sequences. These are given as strings of five characters, A, C, G, T, and N. These characters stand for the four bases (adenosine, cytosine, guanine and thymine), and a character N which indicates that it is not clear which base occupies the given position. Our intention is to extract from such collections of sequences those groups of motifs which share a great level of mutual similarity within the group, but could differ from each other.

2.2 Neighborhood generating mechanism

Standard heuristics require the concept of neighboring solution to the current one in order to ensure an improvement of the current solution. It is necessary to search neighboring solutions before the improved one can be found. In our problem of pattern extraction, let a pattern S_0 (we will call it gene to keep consistency of terminology in this field, although this does not have the same meaning as a biological gene) represents a current solution. Then the neighborhood of S_0 is defined as:

$S_1 \in N(S_0)$, where S_1 can be obtained from S_0 by changing a gene's character on any position to any other allowed gene's character (A, C, G, T).

The operation is illustrated in Figure 1.

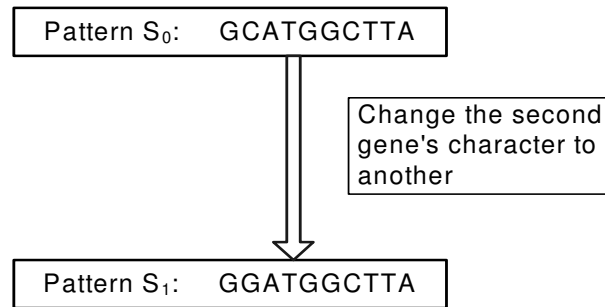


Figure 1: Neighborhood generation

3. OUTLINE OF SIMULATED ANNEALING METHOD

The idea for simulated annealing (SA) based optimization (Kirkpatrick *et al*, 1983) relies on the principles of thermodynamics and resembles the process in which a solid material is first melted and then allowed to cool by slowly reducing temperature. This approach is very suitable for discrete combinatorial optimization problems, such as our problem.

SA searches the attempts to escape local minima by jumping out of them before the solution is too close to local minima. Closer to the end of computation, when the probability of accepting a worse solution is very small, the search simply attempts to find the local minimum similarly as greedy iterative algorithms. We can make a trade off between the ability to find a good solution and to speed up computation. This can be achieved by slowing down the 'cooling schedule'. If the 'cooling schedule' is made slower, this will increase probability of finding the optimum solution. The cost, however, is a longer computation time. Thus, in order to make efficient use of SA we need to find such a 'cooling schedule' which requires acceptable computation time but simultaneously ensures good enough solutions. Thus, the selection of SA parameters is a bit subjective and this represents the main disadvantage. However, the main advantage of SA is its ability to escape local minima. For this reason, its ability to find the global minimum is not related to the initial conditions which represent the initial solution in our case. Another advantage is its very simple implementation. SA is sometimes called a "biased random walk". This due to the fact that iteration steps are made randomly and they do not contain an 'intelligent' move as most of the other optimization techniques. One of the characteristics of SA method is that it does not require the knowledge of the search space. This can be an either advantage or a disadvantage depending on conditions of application and problem in question.

3.1 Implementation of Simulated Annealing

In our implementation, there are three parts of the solution: Controller, RandomRestart and SA. Every time, Controller calls RandomRestart to get an initial solution and passes it to SA. SA runs the initial solution until the criterion function ('temperature') becomes very low and the solution reaches some local minima. Controller then calls RandomRestart again. Figure 2 depicts the flowchart this implementation.

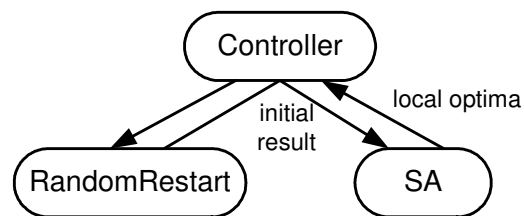


Figure 2. Flowchart of SA implementation

In our SA implementation, for each solution S , we have a *cost* associated with our defined *quality* based on the selected criterion. Usually, the smaller this cost is, the better.

Let us introduce the following annotations:

- r = a random number between 0 and 1;
- T = variable analog to 'temperature' in SA;
- S_0 = current solution;
- S_1 = a neighbor of the current solution;
- $C()$ = the cost of a solution;
- $\Delta(C) = C(S_1) - C(S_0)$;
- $D() = \Delta()$ (within Figure 4);

We can use the relation $r < e^{-\Delta(C)/T}$ to decide whether or not to accept the move from the current solution to the neighbor solution. In the beginning of the process, when T is large enough, these moves are similar to randomly selecting a position and changing the character on this position to another character (i.e. making change of one nucleotide to another which could correspond to mutation event). However, when T becomes very small, and only a downhill move (a move which will reduce the value of the criterion function) can be accepted, we have to find such a downhill move. So, here, we have two problems to solve:

- (a) It is a necessity to have some move order, so that when T is very small we can scan all the possibilities to find a downhill move. Finding a downhill move randomly is not reasonable when T is very small.
- (b) As T is large in the beginning, it is likely that SA will accept most moves no matter are they downhill or uphill. Due to the nature of SA we have to make the move random enough. We cannot always change the gene's character on the first several positions.

To solve the above two problems, we follow the following three steps:

Step1: Randomly reorder all the positions and record down the order R_1 (4, 10, 9, 1, 5, 6, 3, 2, 7, 8).

Step2: Randomly reorder all the gene's characters and record the order R_2 (G, T, A, C)

Step3: For each position in R_1 , change the character on that position to another character in the order R_2 .

These three steps allow us to have a random enough moves and also allow us to have an order to scan all possible moves.

The flowchart of data processing within SA block is shown in Figure 3:

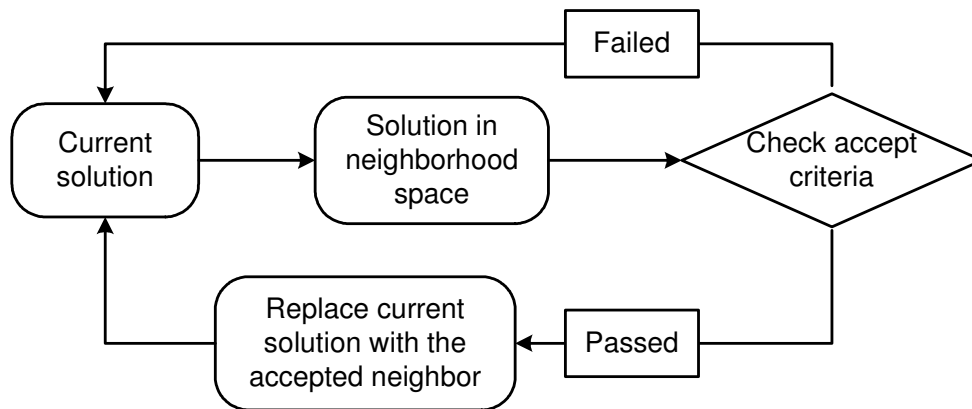
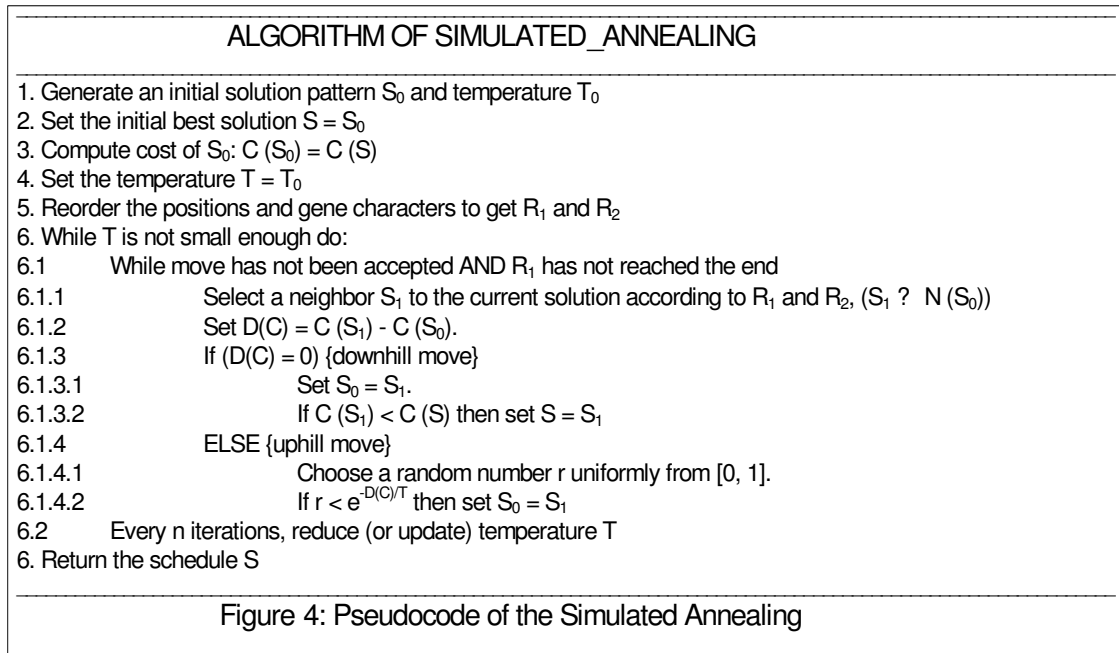


Figure 3. Flowchart of SA

The pseudocode of the SA algorithm is described in Figure 4.



4. TABU SEARCH

Tabu Search is a method introduced by Glover (1986). The overall approach aims at avoiding remaining in iteration cycles by using memory techniques which prevents or penalizes selection of solutions from the points in the solution space which are previously visited.

Four types of memory structures exists in advanced tabu search: Recency-based, Frequency-based, Quality-based, and Influence-based memories. In Recency-based memory we record number of iterations that have passed since a move was executed; in Frequency-based memory we record how often one move has been performed; in Quality-based memory we record the differences of the merit of all solutions visited during the search; Quality-based memory is used to determine elements which are common to good solutions; finally, the Influence-based memory records the effects on quality and structures for all selections made during the search. Using these four types of memories we can realize two key strategies which are part of TS: intensification and diversification.

Intensification strategy modifies choice rules to encourage move combinations and solution features found in past to be good, and thus enables a more thorough search of solution space. Diversification strategy aims to encourage the search process to search unvisited regions and to generate solutions that differ in various significant manners from those already used, and this strategy can radically shift solutions to different section in the solution space.

To avoid retracing the steps used, the method records recent n moves in one or more tabu lists. Any 'tabu' move will be forbidden. However, if a solution we get from a move is the best solution we have, we accept this move even if it is 'tabu', and this is called improved-best aspiration criterion.

4.1 Implementation of TS

There are four blocks in a TS solution: Controller, Intensification and Diversification, Restart and TS. Every time, Controller passes the elite result list containing all the elite results we got so far to Intensification and Diversification. Intensification and Diversification then analyze those elite results to get some constrains for restarting. Restart receives and follows these constrains to get one intensification solution and one diversification solution which will be returned to Controller and will be used as initial solutions in TS in the next iteration. TS searches the neighbors of these two initial solutions and generates a list of elite solution (the function value is within some tolerance). Within TS, some randomly chosen elite results are used as initial solutions to search more thoroughly and TS modifies the elite result list when searching. Lastly, TS returns this elite result list to Controller. Figure 5 shows the flowchart of TS method.

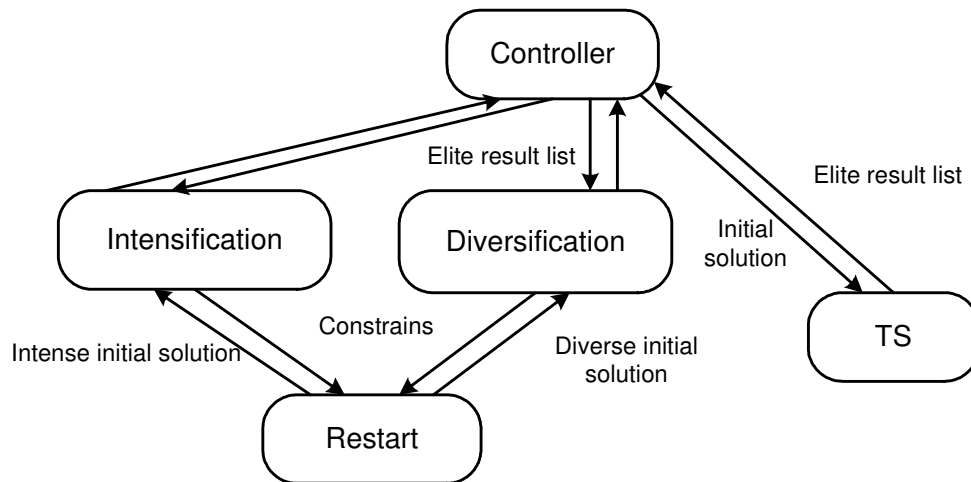


Figure 5: Flowchart of TS implementation

Controller: Controller guides TS to find a good solution more efficiently. It passes information between the Intensification and Diversification parts and the TS part.

Intensification part: This part finds the common elements. Here, the common elements are those which represent the same gene's characters on the same positions in different elite results. To get an intensification result, we still use these common elements.

Diversification part: This part finds the common elements of the elite results. It will not use these common elements.

Restart: This component receives the partial solution from the Intensification and Diversification components to complete it. Here, we use random assignment for non-common positions.

TS: The most important part in TS is Neighborhood Generating Mechanisms. A characteristic of TS different from SA requires that we search all possible neighbors of the current solution and to choose the best non-tabu solution. We also use the improved-best aspiration criterion. In order to use Intensification and Diversification, we also need a memory to record all the elite solutions that are within some defined tolerance to the best solution found so far.

The flowchart inside TS is shown in Figure 6.

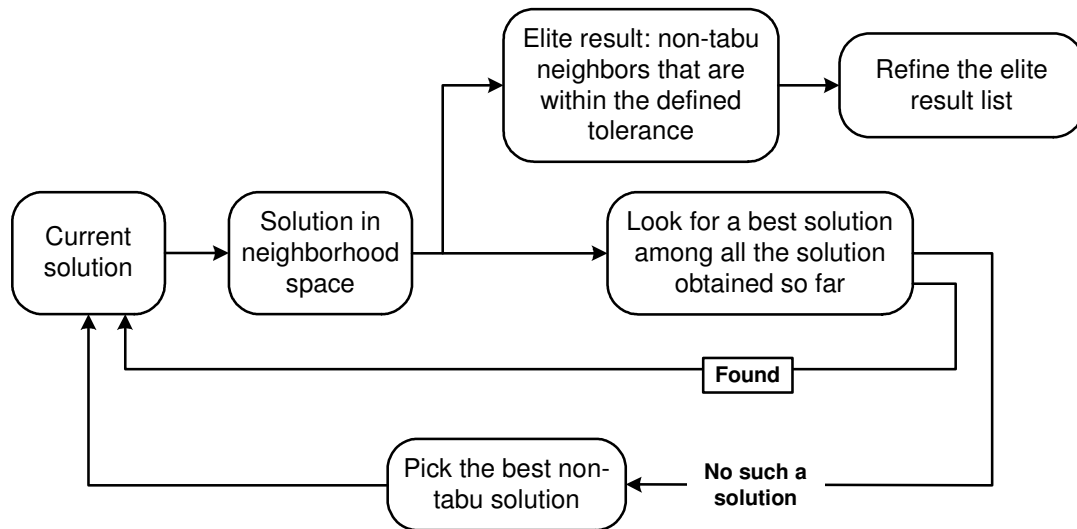


Figure 6: Flowchart inside TS

The pseudocode of Tabu Search is illustrated in Figure 7.

Let us introduce the following annotations first:

- S_0 = initial solution;
- S = current solution;
- S^* = a neighbor of the current solution;

ALGORITHM Tabu_Search

1. Generate an initial schedule S_0 , set the current solution $S = S_0$
 2. Set minimum cost $\min = \text{MAX}$, $\text{non_tabu_min} = \text{MAX}$.
 3. Set $\text{neighbor} = \text{empty}$, $\text{non_tabu_neighbor} = \text{empty}$.
 4. Find the neighborhood space and for each neighbor S^*
 - 4.1 IF $C(S^*) < \min$ THEN
 - 4.1.1 Set $\min = C(S^*)$
 - 4.1.2 Set $\text{neighbor} = S^*$
 - 4.2 IF NOT a tabu move
 - 4.2.1 IF $C(S^*) < \text{non_tabu_min}$
 - 4.2.1.1 Set $\text{non_tabu_min} = C(S^*)$
 - 4.2.1.2 Set $\text{non_tabu_neighbor} = S^*$
 5. IF \min is the best result THEN
 - 5.1 Set $S = \text{neighbor}$ and return to step 2
 6. ELSE
 - 6.1 Set $S = \text{non_tabu_neighbor}$ and return to step 2
-

Figure 7: Pseudocode of Tabu Search

4.3 Time complexity for searching a neighbor in SA and TS algorithms

We have to set the follow three parameters:

K = the number of gene's characters in the chosen pattern;
 M = the number of sequences in the input file;
 N = the number of gene's characters in each sequence in the input file.

1. In SA, for two random reorderings, the time complexity is $O(K)$ and $O(1)$.
2. For neighbourhood move operation, each position has three possible changes, so the time complexity is $O(K)$.
3. To evaluate each neighbor solution, we have to compare it with all the patterns in the input file. For one sequence, there are $N-K+1$ number of patterns, so there is in total $(N-K+1)*M$ patterns in the input file. For each comparison, K characters are compared. So, the time complexity is $O((N-k+1)*M*K) = O(N*M*K)$ (since K is much smaller than N).
4. From 2 and 3, we can get the time complexity of finding a neighbor as $O(N*M*K^2)$.

5. GENETIC ALGORITHM (GA)

GA is based on a population heuristics. It uses a number of current solutions and combines them together to generate new solutions by imitating the genetic process of reproduction. GAs utilize three fundamental principles to create new population: Selection, Crossover and Mutation. Only gene patterns which are most fit will 'reproduce' and create a new population. This is performed in the second step (*Crossover*). The idea behind is that 'good' sections of the parents will combine to produce even more fit children. Although many of the children patterns created in this way will not be sufficiently successful to survive the next selection, some will.

If we repeat these steps then no new solution space can be explored since the previously mentioned two steps make use of the known domains. This may result in a convergence to a local instead of the global minimum. To ensure deviation from the known domains, the 'Mutation' step is necessary. With Mutation a new features are generated not known before and they may be beneficial or detrimental to individual patterns in the population. However, we hope that in a large population some of these mutations will be beneficial. Figure 8 shows the flowchart of Genetic Algorithm.

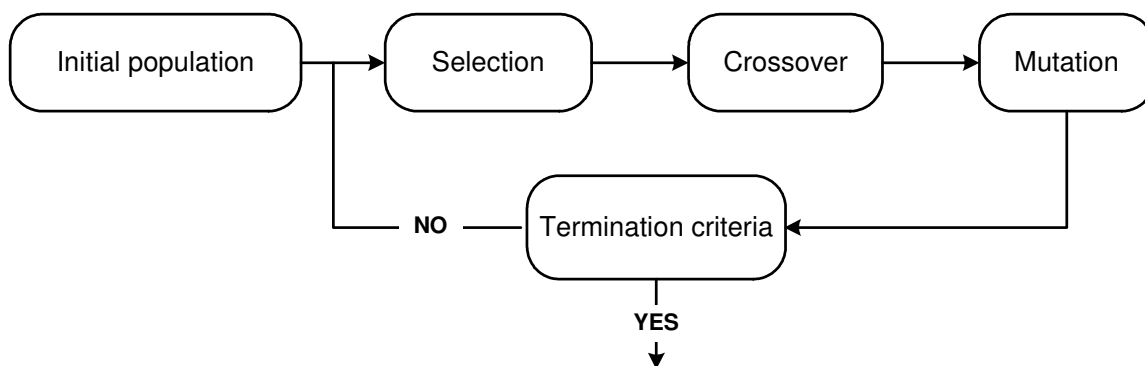


Figure 8: Genetic Algorithm flowchart

5.1 Implementation

Initial Population: The *initial population* of gene motifs is created randomly. The length of the gene pattern is the length of the pattern searched for.

Selection: This block extracts a subset of genes (patterns) from an existing population of patterns, according to the defined fitness. Selection can be performed as described below:

Consider the population where each pattern (gene) has associated fitness. The more fit pattern results in higher fitness value. We calculate the mean-fitness of the population. Every individual pattern will be copied to the new population, at frequency proportional to its fitness (relative to the average fitness). For example, if the average fitness is 5.76, and the fitness of an individual pattern is 20.21, and then we have $20.21/5.76 = 3.51$. This individual pattern will be copied 3 times and also it will have probability of 0.51 to have one more copy in the new population. In our implementation, the size of the population changes dynamically.

Crossover: In our implementation we use a two-point crossover, where we randomly select two positions in parent patterns, cut the parent patterns into three segments and create two children by swapping the segments between the two cutting points. Figure 9 illustrates this strategy.

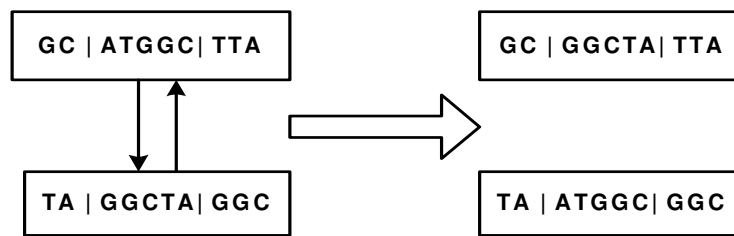


Figure 9: Two points crossover

Mutation: The last step is the Mutation where we use probability P to change any character in a pattern to another character. If the pattern has a length K , the probability of changing to another pattern is $1-(1-P)^K$ (in our implementation we let this value ≈ 0.02).

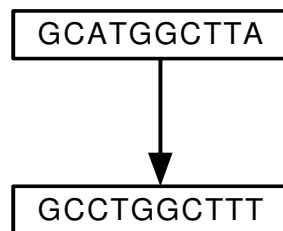


Figure 10: Mutation

6. CONCLUSIONS

We have presented the main ideas in implementation of the simulated annealing, tabu search and a genetic algorithm in the problems of extraction of conserved patterns from a set of unaligned sequences. These algorithms are implemented in a web-accessible tool at http://sdmc.i2r.a-star.edu.sg/DRAGON/Motif_Search/.

REFERENCES

- Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990) Basic local alignment search tool. *J Mol Biol.* **215**(3):403-10.
- Bailey, T.L. and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology (ISMB'94)*, 28-36, AAAI Press, Menlo Park, California, August, 1994.
- Denning, P.J. (1982) Genetic Algorithm, *American Scientist*, **80**:12-14.
- Dowland, K. (1993) Simulated annealing, in *Modern Heuristic Techniques for Combinatorial Problems* (C Reeves, editor), New York, John Wiley & Sons.
- Eglese, R.W. (1990) Simulated annealing: a tool for operational research, *European Journal of Operational Research*, **46**(3):271-281.
- Fleischer, M. (1995) Simulated annealing: past, present, and future, pages: 155 – 161, *ACM Press*, New York, NY, USA
- Glover, F. (1986) Future Paths for Integer Programming and Links to Artificial Intelligence. *Comp. Operations Research*, **13**: 533-549
- Glover, F. (1989) Tabu Search - Part I. *ORSA Journal on Computing*, **1**: 190-206
- Glover, F. (1990) Tabu Search - Part II. *ORSA Journal on Computing*, **2**: 4-32
- Glover, F. and Laguna, M. (1997) *Tabu Search*, Kluwer Academic Publisher

- Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley.
- Hertz, G.Z. and Stormo, G.D. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics.*, **15**(7-8): 563-77.
- Higgins, D., Thompson, J., Gibson, T., Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**:4673-4680.
- Holland, J.H. (1975) *Adaptation in natural and artificial systems*, Ann Arbor: The University of Michigan Press, USA.
- Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schevon, C. (1989) Optimization by Simulated Annealing: An Experimental Evaluation. *Operations Research*, **37**(6): 865-892.
- Kirkpatrick, S, Gelatt, C and Vecchi, M. (1983) Optimization by Simulated Annealing. *Science*, **220**(4598): 671-680.
- Lawrence, Ch.E., Altshul, S.F., Boguski, M.S., Liu, S.L., Neuwald, A.F. and Wootton, J.C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**: 208-214
- Mitchell, M. (1996) *An introduction to Genetic Algorithms*. MIT Press. Cambridge, Massachusetts. London, England.
- Mount, D.W. (2001) *Bioinformatics: Sequence and Genome Analysis*. CSHL Press, USA.
- Pavesi, G., Mereghetti, P., Mauri, G. and Pesole, G. (2004) Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes, *Nucleic Acids Research*, **32**(Web Server issue):W199-W203
- Pearson, W.R. and Lipman, D.J. (1988) Improved Tools for Biological Sequence Comparison. *PNAS*, **85**:2444- 2448.
- Pearson, W.R. (1990) Rapid and Sensitive Sequence Comparison with FASTP and FASTA, *Methods in Enzymology*, **183**:63 - 98.
- Reeves, C.R. (1993) Genetic algorithms. In *Modern Heuristic Techniques for Combinatorial Problems*, 151-196. Oxford: Blackwell Scientific Publications.
- Reeves, C.R. (1997). Genetic algorithms for the Operations Researcher. *INFORMS J. Comp.*, **9**: 231-250
- Roth, F.P., Hughes, J.D., Estep, P.W. and Church, G.M. (1998) Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnol.*, **16**, 939-945
- Tovey, C. (1988) Simulated annealing. *American Journal of Mathematical and Management Sciences*, **8**(3&4): 389-407.
- Werner, T. (1990) Models for prediction and recognition of eukaryotic promoters. *Mamm Genome.*, **10**(2):168-75.

Received: February 20th 2004

Accepted in final format: September 15th 2004 after one revision

About the authors

Liang Yang is a postgraduate student at the Knowledge Extraction Lab, Institute for Infocomm Research, Singapore. He can be reached at yangliang75@hotmail.com

Enli Huang is a researcher at the Knowledge Extraction Lab, Institute for Infocomm Research, Singapore. His research interests are in Bioinformatics and AI. He can be reached at stuhe@i2r.a-star.edu.sg

Vladimir B Bajic is a professor of Bioinformatics and Head of the Knowledge Extraction Lab, Institute for Infocomm, Singapore. His research interests are in Computational methods and AI techniques in Bioinformatics, Nonlinear signal processing, Data mining, Fuzzy Logic, Neural Networks, Stability theory, Systems modeling. He can be reached at bajicv@i2r.a-star.edu.sg