International Journal of Soft Computing 13 (1): 18-24, 2018
ISSN: 1816-9503

Genetic Algorithm for Planted Motif Search (l, d) with Iteration on Average Value

Satarupa Mohanty, Rohan Chauhan and Biswajit Sahoo
School of Computer Engineering, KIIT University, Bhubaneswar, Odisha, India

Abstract: Analysis of motif in DNA sequences are growing into a significant context in the domain of gene regulation and in the same way, identification of motif is a most crucial issue in the study of computational biology. This study addresses the issues of motif search in biological DNA sequences using an evolutionary approach. The experiment is conducted using the Genetic algorithm framework along with a novel approach of generation of a highly fit population of solutions and then converging to optimal motif using secondary fitness function. The proposed model calculate the average value of given sequences and then iterate to generate the optimum population as the strings those are closer to the average value in order to achieve a better rate of convergence to the optimum motif. The experimental study on simulated data using evolutionary approach is recorded and proofs its consistency.

Key words: Planted motif search, Genetic algorithm, average value, convergence, consistency, population

INTRODUCTION

Genes, physical and functional unit of heredity, consist of DNA which are instructions to make molecules called proteins. In genetics, motifs are short signals which are present throughout DNA and are conjectured to have biological significance. Motifs can be of two types namely: Sequence motifs and structural motifs. The sequence motifs are distinguished from sequence motifs based on three-dimensional arrangements of amino acids which may not be adjacent. Motifs play a very significant role in the analysis of gene regulation. Motif discovery is useful in finding binding sites in amino acids, finding regulatory information within DNA or RNA sequences, searching for splicing information and protein domains.

There are three kinds of motif search identified in literature by Hieu Dinh and Sanguthevar Rajasekaran, Planted (l, d) Motif Search (PMS), Simple Motif Search (SMS) and Edit Distance Based Motif Search (EMS). Simple motifs search problem is to identify all the patterns in the given set of sequences such that each pattern is of length at most l, together with a count of how many times each pattern occurs. Optionally a threshold value for the number of occurrences could be supplied. Determining this threshold is a challenging task. A pattern is a string of symbols (also called residues). The length of a pattern is the number of characters in it. The edit distance based motif discovery problem is to find all the patterns in the given set of sequences such that each pattern is of length l and it occurs in at least q of the t sequences. If a pattern V is output, it will mean that V is a string of length l present in one of the input sequences and also, V occurs in at least q of the input sequences. (A string x with $|x| = l$ is considered an occurrence of pattern V as long as the edit distance between x and V is at most d). The planted (l, d) motif search problem is to find a motif (i.e., a sequence) M of length l in the given set of sequences. It is assumed that each input sequence contains a variant of M. The variants of interest are sequences that are at a hamming distance of d from M (Goldberg, 2006).

Literature review: Genetic algorithm (Goldberg, 2006) are search algorithms based on natural selection and natural genetics. Genetic algorithm searches for a solution from a population of points, not a single point. They use objective/fitness function and probabilistic transition rules. Genetic algorithms are commonly used to generate high-quality solutions to optimisation and search problems by relying on bio-inspired operators such as crossover, mutation and selection. Recently several motif discovery methods employing the Genetic algorithm have been perceived by virtue of their convenience towards both global search and local search property. The Genetic algorithm can conquer over the flaws of being converged to the local optimum by performing a global search instead of exhaustive search. Several Genetic algorithms are described as follows:

Liu *et al.* (2004) proposed FMGA algorithm in 2004 which starts with a randomly generated seed population until the predicted motif is unchanged for a fixed number of iterations. In each iteration, FMGA selects the candidates with the highest total fitness score and rest of the candidates are generated using weighted wheel

Corresponding Author: Satarupa Mohanty, School of Computer Engineering, KIIT University, Bhubaneswar, Odisha, India

selection. Once all candidates have been generated, the mutation occurs using calculated weight matrix followed by crossover with ambitious code penalties. The predicted outcomes of FMGA are more exact than the Gibbs sampler.

Unlike FMGA, the GAME Software, addressed by Wei and Jensen (2006) uses two additional operators SHIFT and ADJUST other than the standard Genetic algorithm operators used for motif search problems. GAME employs a Bayesian scoring function for motifs and PWM scan to extract additional motif sites from sequences.The improved version of GAME also works better on unknown motif width.

Bi (2007) presented GEMFA (Genetic based EM Motif-Finding Algorithm) section of Genetic algorithm in 2007 which works as a function optimiser by performing the multiple local alignments in accordance to their maximum likelihoods and minimising the Minimum Distance Length (MDL) value in each iteration as a general principle.

Karaoglu *et al*. (2008) proposed GAMOT in corporates the scoring function as the total distance of the individual consensus. This algorithm is based on the extraction of the best consensus pattern from 4 l possible patterns comprising of highly occurring nucleotide for each motif location. GAMOT reduces the search space to (n-l) t from 4 l consensus.

The operation of MOGAMOD, addressed by Kaya, (2009) is derived from a popular maximal performance multi-objective algorithm NSGA-II (Non-dominated Sorting Genetic Algorithm) (Deb *et al*., 2002). From the non-dominant set of initial solution the MOGAMOD extracts the optimal motif by transforming the optimal motif discovery problem to three diverging optimisation problem as follows: maximisation of similarity, enlarging the motif length and encouragement for candidate motif. The performance of this algorithm is eventually improved by giving the flexibility in the selection of a similarity measure for discovering the motif.

Huo *et al*. (2010) proposed GARPS which integrates the global search ability of the Genetic algorithm with stochastic projection policy of random projection to explore the planted (l, d) motif. The GARPS initially exerts the random projection policy on the input sequences by incorporating the position weight function h(x) and constructing one original hash function h(s) based on position weight function to generate the dense signals with worthy candidates as the starting population for the Genetic algorithm. The Genetic algorithm then starts by making a sequence of iterative processing to refine the candidate motifs.

Fan *et al*. (2015) presented AMDILM, starts with an initial population of the Genetic algorithm by considering the 64 distinct initial individuals or candidates each of length three. Then it employs three operators in sequence as mutation, addition and deletion iteratively on the basis of Total Fitness Score (TFS) value of individual candidates to successively generate the desired length l motif. The total fitness score value of every candidate motif defines their minimum distance value of maximum distances from the given input sequences.

MATERIALS AND METHODS

The whole algorithm is basically divided into three phases. In the first phase, we calculate the average value of n sequences and then in the second phase iterate using the genetic framework employed to generate strings closer to the average value. The first phase of the process is very fast due to the low computational requirement of the fitness function. Once the first phase of the process is complete in the second phase of the process, the 64 generated strings are then used as the seed population. With this seed population in the 3rd phase, the Genetic algorithm is used to converge to the motif.

String representation: In order to identify each l-mer uniquely in the given DNA sequence, we assigned positional values and character values. The character A, G, C, T were assigned 1-4, respectively. Each position in l-mer was given positional value starting from i = 1, ..., l. For example, let's consider a l-mer of length 7 and string be AGTCGTA, then the calculated value of this l-mer would be:

$$\text{Value} = 1\times1+2\times2+4\times3+3\times4+2\times5+4\times6+1\times7 = 54$$

The following approach of having unique character value and positional value results in unique value for each l-mer present in the DNA sequences.

Finding optimum fitness value: On the analysis our first phase of our problem, we observed that the motifs were very close to the average calculated value from each l-mer present in DNA sequences when the weights of each character were taken according to Positional Weight Matrices (PWM) (Neil and Pevzner, 2004). However, this led to a problem of building up the motif from the calculated average value, taking back into consideration the PWM and positional value, making it an exponential problem in itself. Hence, in order to generate a high fitness population, we adopted the representation and developed an algorithm for finding seed population with high fitness.

Genetic algorithms were best suited for the purpose of generating candidate motifs closer to the calculated average value due to their ability to build better and better strings from the best partial solutions of past samplings.

Fitness function: We have used two fitness functions: primary and secondary. In the second phase using the primary fitness function, we iteratively find candidate l-mers for the Genetic algorithm and in the third phase using secondary fitness function, we find the motif.

Primary fitness function: To find the motif faster using the Genetic algorithm, we initialise the evolutionary process with a high fitness population using primary fitness function. The primary fitness function provides a new methodology to generate the high fitness population by generating population closer to average value using the definitions as:

Definition (Eq. 1): We consider the fitness score of a given candidate l-mer C^l. For a given l-mer C^l, the fitness score is calculated by fitness score function, defined as follows:

$$FS(C^l)=\sum_{n=1}^{l} \text{value}(C_i, i), C_i \in \{A, G, C, T\} \text{ and } i = 1,2,3, ..., l$$

Where:

$$\text{value} = (C_i, i)\begin{cases} i*1, & \text{if } C_i = 'A' \\ i*2, & \text{if } C_i = 'G' \\ i*3, & \text{if } C_i = 'C' \\ i*4, & \text{if } C_i = 'T' \end{cases} \quad (1)$$

Where:

'i' = The ith position

C_i = Character at ith position

FS gives us the summation of these values to give the unique value of l-mer. The score of l-mer calculated using definition 1 and is used iteratively in algorithm 2 to generate the candidate population using the definition as:

Definition (Eq. 2): Once we have obtained the fitness score FS of l-mer according to the proposed representation, we need to find out how far we are from the average value obtained using Algorithm 1. The closer the string is to the average value, the fitter it is. Finally, the primary fitness function is defined as follows:

$$S(C^l)=\left|a - FS(C^l)\right| \quad (2)$$

Where:

'a' = The average value calculated from all l-mer in n sequence

'C^l' = A given l-mer the lower value

S = Implies higher fitness score

Secondary fitness function: The candidate l-mers obtained from Algorithm 2 are initialised as the seed population for the Genetic algorithm which uses the secondary fitness function to converge to the motif. The secondary fitness function is defined as follows.

Definition (Eq. 3): For a sequence $S^i \in \{S_1, S_2, ..., S_N\}$ the secondary fitness score of a candidate motif C^l of length l is given by the Hamming distance of l-mer with at most d mismatches (H_d). In case, more than d-mismatches occur, the least Hamming distance of candidate motif is taken as the fitness score for that sequence. In order to improve the computation time, we move on to the next sequence as soon as l-mer with at most d mismatches to motif is encountered. The fitness score calculated by secondary fitness score function is defined as follows:

$$F(C^l_i, S_t) = \min(H_d(C^l_i, S_t)), \quad C_i \in \{C_1, C_2, ..., C_{64}\} \text{ and } S_i \in \{S_1, S_2, ..., S_N\} \quad (3)$$

The score calculated above using definition 3 is used below to obtain the final fitness value of the candidate l-mer.

Definition (Eq. 4): For a candidate motif C_i, the total secondary fitness score with respect to all sequences $S = (S_1, S_2, ..., S_N)$ is given by the fitness function defined as follows:

$$TFS(C_i, S)=\sum_{n=1}^{N} F(C^l_i, S_n)$$

Where:

M = The total number of sequences

C^l_i = The candidate motif. Lower the T F S, better is the fitness of candidate motif

Operators: In solving Eq. 1, mutation, addition and deletion operators are used as mentioned in AMDILM (Fan *et al*., 2015). Traditional Mutation operator is used in changing any random site of motif into one of A, G, C, T keeping in mind that the original 3 sites of the initial population remain unchanged. Changing these original 3 sites would otherwise reduce the diversity of population to identical strings.

Given a candidate motif $M = \{M_1, M_2, ..., M_L\}$ of length L, the addition operator randomly selects a nucleotide from A, G, C, T and is added to the last position to make $M' = \{M_1, M_2, ..., M_L, M_{L+1}\}$ and to the initial position to make $M'' = \{M_0, M_1, M_2, ..., M_L\}$. After that both M' and M'' are scored using the fitness function given in 1 and the superior one is reserved. Deletion operator deletes the last added nucleotide from M_{L+1} to reverse it back to M_L.

In solving Eq. 2, we used the fitness function defined in definition 4 to evaluate the 64 candidate l-mers obtained from Alogrithm 2. Traditional mutation and crossover operators are used with a mutation probability of 0.2.

Program implementation: The proposed Genetic alogrithm framework is divivded into three phases. In the first phase, Alogrithm 1 computes the average value of the given n sequences and then in the second phase, Alogrithm 2 iterates to bring out the highly fit population those are closer to the computed average value. Finally, in the third phase, Alogrithm 3 employs the Genetic alogrithm to converge the strong candidate l-mers to the optimal motif.

Alogrithm 1; Calculate average value:

Input: A file f containing 20 DNA sequences embedded with motif and l is the length of l-mers

Output: It returns the average value calculated from all l-mers in the file.

```
Function average value (l, file)
  for each sequence in file do
    calc-sum ←0
    for each l-mer in sequencr do
      pos ← 1
    for each char in l-mer do
      if char = 'A' then
        u ← pos* 1
      else if char = 'G' then
        u ← pos 2*
      else if char = 'C' then
        u ← pos 3*
      else char = 'T'
       u ← pos 4*
      end if
     pos ← pos + 1
     calc-sum ← calc-sum+u
    end for
    calc-sum ← calc-sum/(600-l)
    avg-calc-sum ← avg-calc-sum+calc-sum
  end for
 end for
av avg-calc-sum ← avg-calc-sum/20
return avg-calc-sum
end function
```

In Algorithm 1, the average value is calculated from n sequences given in the file. For each sequence, the algorithm calculates all l-mer's value using the string representation given in 1 and aggregate these to compute the average value for each sequence by dividing the aggregate with the total numbers of l-mers present in the sequence. Then finally, it figures out the final average value by evaluating the average of all the averages from all sequences.

Algorithm 2; Generate candidate motif set:

Input: Average value a obtained from function, max is number of times to try for diverse motifs and l the length of l-mer.

Output: It returns a set containing 64 l-mers which are candidates to be motif.

Initialisation: $\{C^0_1, C^0_2, ..., C^0_{64}\}$ are initial set of candidates, compute the fitness score $\{S_1, S_2, ..., S_{64}\}$

```
function population generator (a, l, max)
  x ← 3
while x≤l do
  for i = 1: 64 do
    get C_i1^x+1 by using Addition operator on C_i^x
    compute the fitness score S_i1^x+1 from C_i1^x+1
    for y = 1: max do
      get C_i1^x by using Deletion operator on C_i1^x+1
      if x > 3 then
        get C_i2^x by using Mutation operator on C_i1^x
        get C_i2^x+1 by using Addition operator on C_i2^x
      else
        get C_i2^x+1 by using Addition operator on C_i1^x
      end if
      compute the fitness score S_i2^x+1 from C_i2^x+1
      if S_i1^x+1 < S_i2^x+1 then
        S_i1^x+1 ← S_i2^x+1
        C_i1^x+1 ← C_i2^x+1
    end if
  end for
  S_i^x+1 ← S_i1^x+1
  C_i^x+1 ← C_i1^x+1
 end for
 x = x + 1
end while
return C (set of candidates {C^l_1, C^l_2, ..., C_64})
end function
```

Algorithm 2 begins by considering the initial length of the individual population as three. For length three and character set {A, G, C, T}, using the permutation gives us a total of 64 distinct individuals in the initial population. We have taken 64 candidates because it is the set of all possible population of length three which precludes any chances of not finding the motif by making the search global. In each iteration, the algorithm exerts three operators mutation, addition and deletion on the candidates to increase the length of the candidate until we reach up to the desired length of the motif with optimal candidate motif. The nucleotide to be added to each candidate is decided by the fitness function given in 1.The variable max decides the number of trials to be performed (value ranging from 3-5) before it moves on to next candidate and serves to increase the diversity of solution.

Algorithm 3; Find motif:

Input: A file f containing 20 DNA sequences embedded with motif, l is the length of l-mer, d is the number of allowed mismatch, max is number of times to try for diverse motifs and m is the mutation probability.

Output: It returns the motif.

```
function find motif (f, l, d, m, max)
```

```
    a ← averageV alue (l, f)
    s ← population generator (a, l, max)
    Population_i ← s
    while motif is not found do
        evaluate populationi using fitness function
        get Z_i by applying Crossover operator on population_i
        apply mutation operator on Z_i
        i ← i + 1
        Update population_i ← Z_i
    end while
    return best (population i)
end function
```

In Algorithm 3, a basic Genetic algorithm is employed to reach the exact solution. The 64 candidatel-mers obtained as a result of algorithm 2 are initialised as the seed population. Then crossover and mutation operators are applied on the population until we converge to the solution. We use one point crossover with a mutation probability of 0.2. The flow chart in Fig. 1 represents logical view of the proposed algorithm.

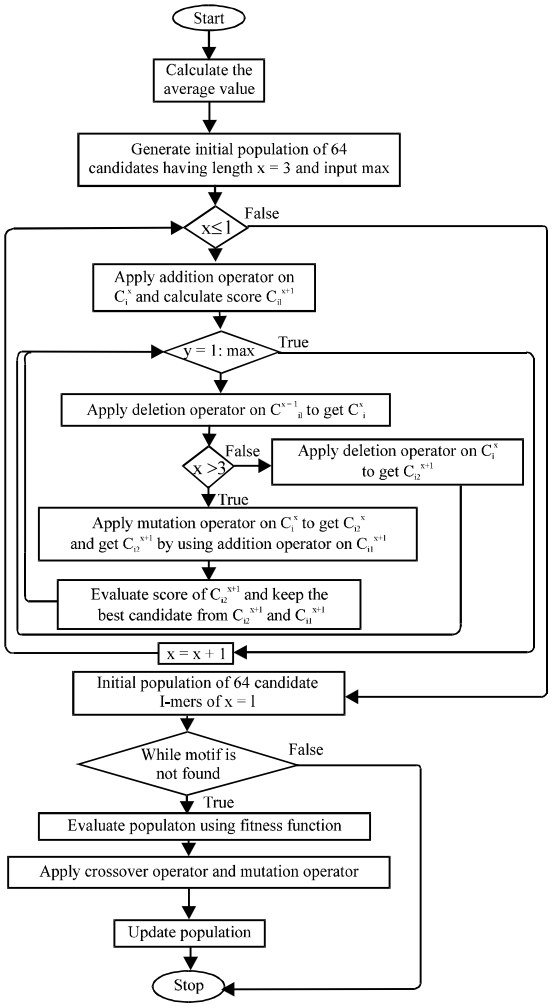

Fig. 1: Flowchart of the proposed algorithm

RESULTS AND DISCUSSION

We implemented the algorithm using C++. The datasets used to evaluate the algorithm are obtained from Nicolae and Rajasekaran (2014). The dataset contained 20 random DNA sequences, each of length 600 as the input instances to experimentally evaluate our algorithm. For every (l, d) combination we report the average runtime over 5 random instances. We compare our results with few of the recent algorithms.

Figure 2 and 3 shows graphs in which difference between the calculated average value and original motif value varies with the length of the l-mers. We see that the graph is full of irregularities and as such no pattern is evident from the graph itself, however, the differences remain small. Although, the differences seem to increase with the length of l-mers, an increase in the length of the l-mern compensates for the fact to the number of changes to be performed in the average string considering the string representation given in 1.

Table 1 shows the efficiency of the proposed algorithm in terms of execution time for different l and d values in comparison to GAMOT. Figure 4 shows the plot of the running times.

Performance coefficient is taken as $|K \cap P|/|K \cap P|$ where K is the set of known motif positions and P is the set of predicted positions in the given instances. The proposed algorithm has a performance coefficient of 1 as it always finds out the exact solution. To show the quality of solution in proposed algorithm, we compare the proposed algorithm with PROJECTION (Buhler and Tompa, 2002), VAS (Chin and Leung, 2006) and GARPS in Table 2. Figure 5 shows the graph of performance coefficient for different l values.

Table 1: Comparison with GAMOT

l	d	GAMOT (sec)	Proposed algorithm (sec)
8	2	13	10
10	2	15	13
12	3	12	11
14	4	10	10
16	5	13	12
18	6	14	11
20	7	27	23
30	11	59	52

Table 2: Performance coefficient comparison

l	d	GARPS	Projection	VAS	Proposed algorithm
10	2	0.906	0.67	0.86	1
11	2	0.992	0.90	0.18	1
12	3	0.805	0.90	0.83	1
13	3	0.963	0.82	0.87	1
14	4	0.873	0.90	na	1
15	4	0.947	1.00	1.00	1
16	5	0.725	0.74	0.65	1
17	5	0.947	0.74	0.94	1
18	6	0.814	1.00	0.86	1
20	7	-	-	0.73	1
30	11	-	-	1.00	1
40	15	-	-	1.00	1

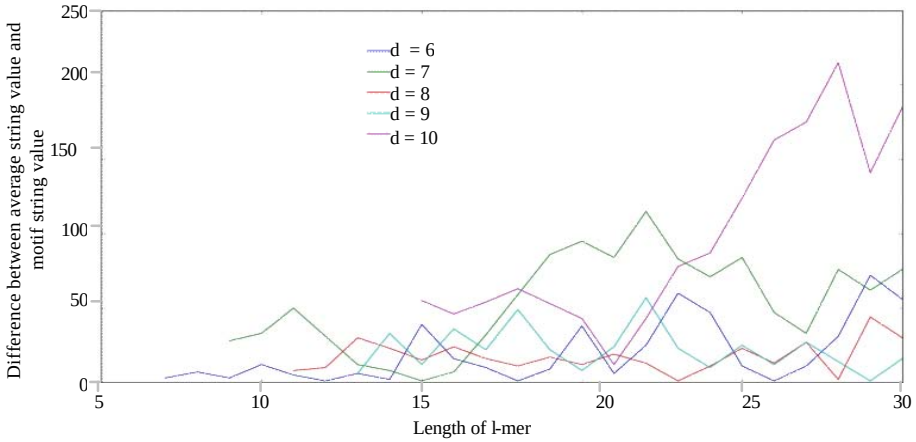

Fig. 2: Graph showing absolute difference between the original motif's calculated value and average string calculated value for d = 1 to d = 5

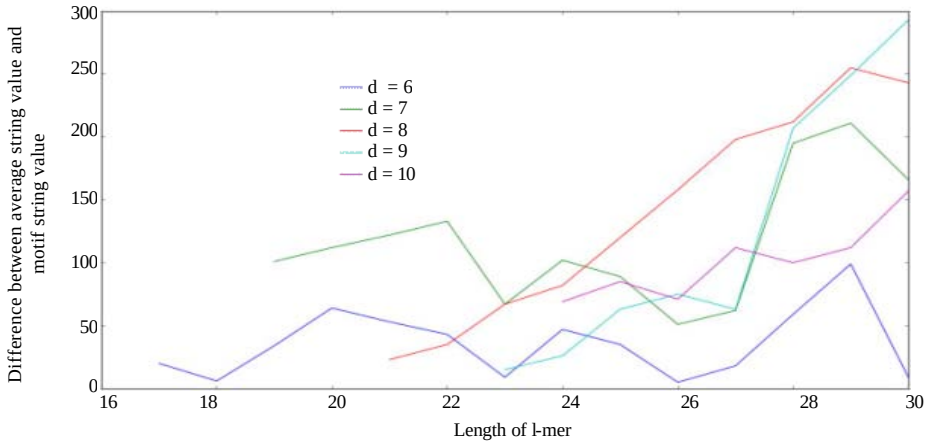

Fig. 3: Graph showing absolute difference between the original motif's calculated value and average string calculated value for d = 6 to d = 10

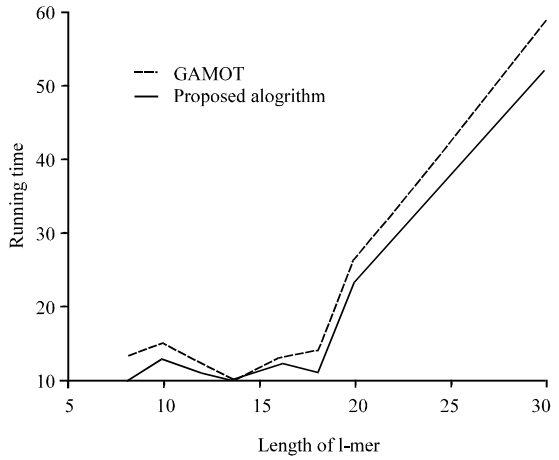

Fig. 4: Graph of running time comparison between proposed algorithm and GAMOT

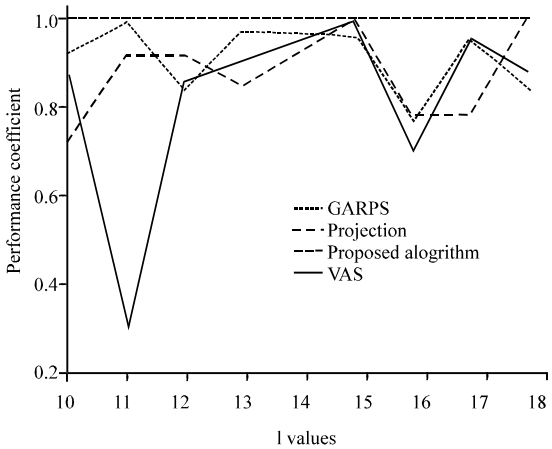

Fig. 5: Graph of performance coefficient comparison

CONCLUSION

In this study, we presented a new genetic framework to find the solution of planted motif search (l, d) which first find a highly fit population closer to the global average and then uses this seed population to converge to the motif. The seed population obtained using the average value, provides very strong signals and can easily converge to the global optimum with the assistance of global search potentiality of Genetic algorithm. The Genetic algorithm can be fine tuned as per different parameters for optimising the solutions to both challenging instances and non-challenging instances of the problem. Although, the algorithm, doesn't generate all the motifs starting from an initial length up to the required l-mer, the algorithm is faster at finding motifs and solving challenging instances of PMS problem.

REFERENCES

Bi, C., 2007. A genetic-based EM motif-finding algorithm for biological sequence analysis. Proceedings of the IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, April 1-5, 2007, IEEE, New York, USA., ISBN:1-4244-0710-9, pp: 275-282.

Buhler, J. and M. Tompa, 2002. Finding motifs using random projections. J. Comput. Biol., 9: 225-242.

Chin, F.Y. and H.C. Leung, 2006. An Efficient Algorithm for String Motif Discovery. In: Proceedings of the 4th Asia Pacific Bioinformatics Conference, Jiang, T., U.C. Yang, Y.P.P. Chen and L. Wong (Eds.). Imperial College Press, London, UK., pp: 79-88.

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, 2002. A fast and elitist multiobjective Genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput., 6: 182-197.

Fan, Y., W. Wu, J. Yang, W. Yang and R. Liu, 2015. An algorithm for motif discovery with iteration on lengths of motifs. IEEE./ACM. Transac. Comput. Biol. Bioinf., 12: 136-141.

Goldberg, D.E., 2006. Genetic Algorithms. Pearson Education, India,.

Huo, H., Z. Zhao, V. Stojkovic and L. Liu, 2010. Optimizing Genetic algorithm for motif discovery. Math. Comp. Modell., 52: 2011-2020.

Karaoglu, N., S.M. Stroh and B. Manderick, 2008. Gamot: An efficient ge-netic algorithm for finding challenging motifs in DNA sequences. Regul. Genomics, 8: 43-43.

Kaya, M., 2009. MOGAMOD: Multi-objective Genetic algorithm for motif discovery. Expert Syst. Appl., 36: 1039-1047.

Liu, F.F., J.J. Tsai, R.M. Chen, S.N. Chen and S.H. Shih, 2004. FMGA: finding motifs by Genetic algorithm. Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering, May 21, 2004, IEEE, New York, USA., ISBN: 0-7695-2173-8, pp: 459-466.

Neil, C.J. and P. Pevzner, 2004. An Introduction to Bioinformatics Algo-Rithms. MIT Press, Cambridge, Massachusetts, USA.,.

Nicolae, M. and S. Rajasekaran, 2014. Efficient sequential and parallel algorithms for planted motif search. BMC. Bioinf., 15: 1-34.

Wei, Z. and S.T. Jensen, 2006. GAME: Detecting cis-regulatory elements using a Genetic algorithm. Bioinf., 22: 1577-1584.