Genetic Algorithm Based Approach for Obtaining Alignment of Multiple Sequences

Ruchi Gupta[1]
Ph.D Research Scholar MCA Deptt
AKGEC Ghaziabad

Dr. Pankaj Agarwal[2]
Professor & Head, Department of
Computer Science & Engineering
IMS Engineering College Ghaziabad

Dr. A. K. Soni[3]
lProfessor & Head, Department of
Computer Science & Engineering
Sharda University, Greater Noida

***Abstract*—This paper presents genetic algorithm based solution for determing alignment of multiple molecular sequences. Two datasets from DNA families Canis_familiaris and galaxy dataset have been considered for experimental work & analysis. Genetic operators like cross over rate, mutation rate can be defined by the user. Experiments & observations were recorded w.r.t variable parameters like fixed population size vs variable number of generations & vice versa, variable crossover & mutation rates. Comparative evaluation in terms of measure of fitness accuracy is also carried out w.r.t existing MSA tools like Maft, Kalign. Experimental results show that the proposed solution does offer better fitness accuracy rates.**

Keywords-DNA Sequences; alignment; Genetic Algorithm; Crossover; Mutation; Selection; Multiple Sequence Alignment etc.

I. INTRODUCTION

Simultaneous alignment of several sequences is among the most important problems in computational molecular biology. Multiple sequence alignment (MSA) can be seen as a generalization of Pairwise Sequence Alignment where instead of aligning two sequences, n sequences are aligned simultaneously, where n is > 2. Multiple sequence alignment can discover biologically significant sequence patterns that may be widely dispersed or hidden in the molecular sequence databases. MSA gives insight into the basis for sequence of similarities between homologous sequences. [1]

An example of an alignment of four hypothetical DNA sequences is shown in Fig. 1.

```
- G C T G A T A T A G C T
G G G T G A T - T A G C T
- G C T - A T - - C G C -
A G C G G A - A C A C C T
```

Figure1: An Example of an Alignment

The basic idea is that the sequences are aligned on top of each other, so that a coordinate system is set up, where each row is the sequence for one protein, and each column is the 'same' position in each sequence. Each column corresponds to a specific residue in the 'prototypical' protein.

Multiple Sequence Alignment (MSA) is considered to be an important tool for computational biologists. It finds its application in phylogenetic analysis, identification of conserved motifs and domains and structure prediction [3]. MSA is a computationally difficult problem, also known to be a NP-hard problem [2]. Considering both the importance and complexity of solving the MSA problem, many different heuristic methods have been proposed by the researchers to provide approximate solutions to this problem.

Genetic Algorithms (GAs) as a computational means to solve the MSA problem has shown lot of potential. It can search through the solution space effectively and generate good alignment results. The main advantage of genetic algorithms over other optimization methods is that there is no need to provide a particular algorithm to solve a given problem. It only needs a fitness function to evaluate the quality of different solutions. Also since it is an implicitly parallel technique, it can be implemented very effectively on powerful parallel computers to solve exceptionally demanding large-scale problems.

The method works by breaking a series of possible MSAs into fragments and repeatedly rearranging those fragments with the introduction of gaps at varying positions. This paper also explores the possibility of applying GA based solution for MSA problem. One such proposed & developed solution is also presented.

II. RELATED STUDY

Genetic algorithm is one of the useful tools determining alignment of multiple sequences. Iterative methods may be implemented through evolutionary approach that use computational heuristics based on natural biological phenomena such as selection, crossover and mutation to evolve a population of candidate solutions based on an objective function because they work similarly to progressive methods but repeatedly realign the initial sequences as well as adding new sequences to the growing MSA [3].

There are some proposed iterative methods to improve the problem of MSA. For example, evolutionary approach SAGA [5] based on genetic algorithm have been success fully applied to the MSA problem. It is used to optimize two different objective functions and shows that they can search large solution space efficiently. But due to repeated use of fitness function it may increase its time complexity.

Zhang C et al., [7] proposed an algorithm based on genetic algorithm and dynamic programming. It was used with two different distance matrices and characterized by great

complexity in processing time. It has some limitations for performing crossover and mutation operations.

One of the most appropriate GA approaches to solve the MSA problem was presented by Nguyen et. al [8], however there are still some limitations w.r.t scoring scheme.

Another useful algorithm for multiple DNA sequence alignment using genetic algorithms and divide-and-conquer techniques [9] was proposed in which optimal cut points of multiple DNA sequences were selected. According to the author experimental results showed quite significant results. Approach involves cutting of the sequences for decreasing the space complexity for sequence alignment. However alignment was possible only for multiple deoxyribonucleic acid sequences, not for protein and other nucleic acid sequences.

Other new genetic algorithms [10] were used for solving the MSA in which various dataset were tested and the experimental results were compared with other methods. But after comparison it was observed that this approach could obtain good performance in the data sets with high similarity and long sequences.

After that effective GARS approach [11] based on Genetic Algorithm with Reverse Selection was proposed. But it suffers from premature convergence in which solution reaches locally at an optimal stage. Furthermore a new approach AlineaGA [12] was proposed which used a

Genetic Algorithm with local search optimization embedded on its mutation operators for performing multiple sequence alignment. But its mutation probability leads to better solutions in fewer generations and that the mutation operators had a dramatic effect in this particular domain. Recently a new Cyclic Genetic Approach (CGA) [13] developed with the complete knowledge of the problem and its parameters. In CGA, the values of various parameters are decided based on the problem and fitness value obtained in each generation. But the column score value varies for each execution may not give relatively better alignment.

In this paper, we proposed an evolutionary approach using genetic algorithms to obtain alignments of multiple sequences. Experimental results show that the proposed solution does offer better fitness accuracy rates w.r.t some existing tools.

Methodology

The remainder of this section is organized as follows. . In section 3 we present genetic algorithm based approach (GAMS) for solving the problem of aligning multiple sequences. Section 4 shows the experimental results of various dataset which are used to test the performance of our method. Then section 5 is finally used for discussion and conclusion.

III. GENETIC ALGORITHM BASED APPROACH

In this section we present our algorithm for solving the MSA problem. Genetic algorithms based approach (GAMS) are applied with new selection and crossover scheme which helps us to generate best population on local schema so that better alignment could be discovered. This process flow is depicted in figure 2.

```
GA_MS ()
// initialize a usually random population of individuals
        initpopulation P (t);
// evaluate fitness of all initial individuals in population
        Evaluate P (t);
// test for termination criterion (fitness core)
 While (not find best solution)
{
FOR i = 1 TO n DO
{
 // Select two chromosomes X and Y with highest fitness value from current evaluation
        P' := select parents P (t);
// recombine the "genes" of selected parents
        Recombine P' (t);
// perturb the mated population stochastically
            Mutate P' (t);
 // evaluate its new fitness
            Evaluate P' (t);
 // select the survivors from actual fitness
            P := survive P,P' (t);
        Od
       }
   end GA.
}
```

Figure 2: GA Process flow

A. Chromosome Representation

The chromosome should in some way that contains information about solution which it represents. The most used way of encoding is a binary string. The chromosome then could look like this: Each chromosome has one binary string. Each bit in this string can represent some characteristic of the solution or the whole string can represent a number. Of course, there are many other ways of encoding. This depends mainly on the solved problem. For example, one can encode directly integer or real numbers; sometimes it is useful to encode some permutations and so on. Each sequence has its own length. The number of gaps in the sequence is to be inserted in each sequence. It is calculated in a way that the length of all sequence remains the same. Therefore we have to generate the maximum length of sequence by multiplying the maximum length of particular element of sequences with rsp1.2. Let's say we have a set of sequence S = {S1, S2, S3 ….Sn}. So the maximum length of the column has to be found out by multiplying the sequence with rsp by maximum length column. The value of scaling factor rsp defines that the alignment to be 20% longer then the sequence which is based on the

observations that solution to common MSA problem really contains more than 20% gaps. The flow of chromosome representation is shown in figure 3:-

```
Create_Random_Matrix(n,L)
{
//generate the initial population G
//let n be size of Population
FOR  G = 1 TO n DO
{
Select Length of Input Sequence=L
For  i=1 to L DO
                                        {
//Create Position of Random Spaces
//Generate minimum value is 1
//Generate Position of Space
// define the value of scaling factor=1.2
 Position = (RandomSpaces(min, L + 1));
}
}
}
```

Figure 3: chromosome representation

B. Evaluation of Fitness Function

To evaluate their fitness, the chromosomes must be converted to the alignment form to be applied sum-of-pairs function [3]. We scored each column by looking at matches, mismatches, and gaps in the two sequences. We assume that a match = 1, a mismatch = 0, and a gap = -1. The fitness or scoring function of each individual is calculated by the formula:-

$$Fitness\_Score = \sum_{i=1}^{p-1} \sum_{j=j+1}^{p} ScoringMatrix(A_i, A_j)$$

The fitness Score for each alignment is calculated by summing the individual score for each column in the matrix. Scoring matrix is needed to determine the cost of aligning a residue with another. Also, a gap penalty value must be settled for determining the cost of aligning an amino acid with a gap. This penalty is only employed when aligning a residue with a gap. The fitness value calculation is to be represented by figure 4:-

```
FitnessValue(G, Max_length)
{
//generate G is the Gap penalty
//Calculate Sequence Count
//Calculate max_length of Sequences
for(a=0 to (max_length+a))
{
//check position of sequence in matrix not null or  ' - ' fitnessvalue+=0
//check position of sequence in matrix  null or '-''
fitnessvalue += G
}
}
```

Figure 4: The flow for finding best fitness score

C. Selection Procedure

After calculating the fitness score of all the population applying larger tournament method where n individuals are randomly chosen, the fitter of the two is selecting with the highest and second highest fitness value .In this case the fitter the individual is chosen by the following procedure:-

- Apply larger tournament strategy for the current population based on their fitness function
- Select two best chromosomes randomly based on their column score and select two individual with their highest fitness value.

D. Crossover

In the single point crossover process, Crossover selects sequence from parent chromosomes and creates a new offspring. The simplest way how to do this is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent .we select crossover point at the rate of 0.5 and count the entire gap in each population then multiply it with crossover rate and take ceiling of crossover rate. The crossover point is selected by the formula:-

$$Crossover\ldots point = total\ldots no\ldots of\cdots gaps \times 0.5$$

After selecting point, copy the chromosome of first parent exact at the crossover point value then copy all chromosome of second parent and vice versa so [13]. There are two offspring has to be generated after applying the crossover function. Calculate the fitness score of current population and select the best individual for performing mutation operation. The flow of one point crossover is shown in figure 5.

```
CrossOver (b,p)
{
//let cr be the cross rate
//let crosspoint be the cp
//cp=sequencecount+cr
//pick an array b[] in the range crossover from random
//Declare p as point =b[]
for i=1 to  Do cp
{
```

```
if b[i]=p[p+cp]Go to first step
else
b[p+cp]=p[p+cp-i]
}
//let position of matrix pos
pos=b[i]
if b[i]>max_length
max_length +=max_length
pos=pos-(max_length*(sequencecount-1))
 }
```

Figure 5: flow of one point crossover

E. Mutation

After a crossover is performed, mutation takes place. This is to prevent falling all solutions in population into a local optimum of solved problem. The system randomly chooses a gene of a chromosome form the mating pool randomly and applying binary mutation. Mutation changes randomly the new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1 [9]. where all the gaps are represented by 0's and all the base symbols are represented by 1's and mutation takes place separately in each sequence up to the mutation point rate of 0.2 [9] is initialized and corresponding mutation point is selected. The mutation point is to be selected by the formula:-

$$Mutation\ldots point = 0.2 \times total\ldots length\ldots of\ldots bit\cdots string$$

First the mutation operator converts the total sequence in to bit string then calculate the mutation point after calculating the mutation point every picks a random amino acid from a randomly chosen row (sequence) in the alignment and checks whether one of its neighbors has a gap. If this is the case, the algorithms swap the symbols. The flow of one point crossover is shown in figure 6.

```
Mutation (mp,i)
{
//Declare mutation point mp
//declare sequence row count count
mp=count* mutation rate
//declare string symbol
 for i=1 to mp do
{
symbol=removeSymbols.SubString(i,1)
 if(matrix row='-')
matrix row =symbol
else
 matrix row = '-'
}
}
```

Figure 6: flow of space mutation

IV. IMPLEMENTATION AND RESULTS

The algorithm is implemented using Microsoft visual studio and the machine for this research is a personnel computer with Intel Pentium III processor .The main memory is 4 gigabyte and Microsoft XP was used as a platform for the implementation. The DNA query input sequence is to be taken from cans family. Query input format of the DNA sequence is listed in figure 7:-

>SPAC1002.14|1824570|1826248|itt1|I

GTAAATTCATACCGGAAATTTTACCAAATGGCGAT
TTCTTAATTGCTGAGGTGGCCAGCAGAAATCGTCTTT
TCATTATTCTGGAATCAAAACACATTCTTTGAATTGTT
CACTTTTCTGTTGCCTTGAAATCTTGGTCTTCTTAGTT
GACTGTTTCATCAAGGTTGCTCCAAATTCTTTGTGATT
TATTGGTAAACTCGGGCATTTTATTGAGT

>SPAC10F6.10|1225464|1227365|SPAC10F6.10|I

TTTTTATATACCAGTTTTATTTACAACAAAAAGTTT
TTACTACCACCTACAAAATACAAAAACTTGGATTTGT
ATCCAGTTCTTTGTCAAATTTTTAAATAAATTATTCTT
TTATTGATTTATTTAAAGTTTAAG

Figure 7: Query Input Formats

During the course of experiments, we have tried various chromosome lengths in order to understand how they have an effect on the performance of the GA.

Datasets

We have used two datasets which are DNA sequences from two DNA families, Canis_familiaris dataset (psm3, SPAC105.03c, taf11, SPAC1142.01) and galaxy dataset (AY395516.1,AY390420.1,AY390421.1,AY390422).These datasets are used as input to our multiple sequence alignment. The parameters setting for experiment are summarized in table 1.

Parameter	Content
Population size	5,10,15,20
Generation	50,100,150,200
Selection Strategy	Random Selection
Crossover operator	One Point
Crossover Rate(Rc)	0.8,0.6,0.3
Mutation operator	Space Mutation
Mutation Rate(Rm)	0.5,0.3,0.1
Scoring Matrix	Scoring 1
Gap penalty	-1
Accumulate size	20%

Table 1: GA Parameters

In order to examine our algorithm validity, we test number of series with DNA sequence. Firstly the algorithm is executed 100 run on GA while number of generation become fix and size of population is to be continued changed. It is observed that running time is increased accordingly and indicate by a notable rise in fitness score about 10% after increasing each size of population. Again while size of population become fixed and number of generation continued to changed then it has to be notified that running time is increased accordingly and indicate by a notable rise in fitness score about 10% . Table 2, 3 and figure 8, 9 lists the results after varying the number of generation and size population.

Size of Pop	No of Gen	Fitness Score	Running Time
5	100	-831	2:06
10	100	-931	4:78
15	100	-940	8:20
20	100	-990	12:08

Table 2: Operators assemble on 5, 10, 15 and 20 size of population with fix number of generation with and calculated fitness score, running time

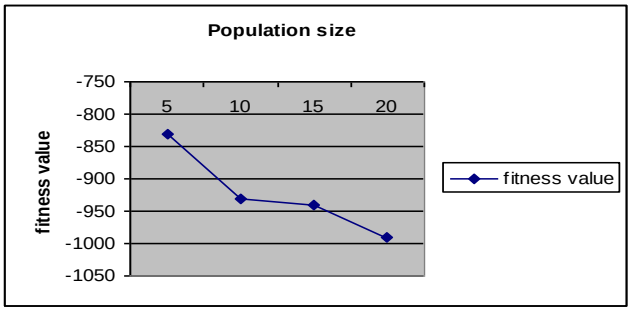

Figure8: Fitness curve GA with Verifying size of population

Size of Pop	No of Gen	Fitness Score	Running Time
5	50	-831	1:03
5	100	-889	2:05
5	150	-909	3:07
5	200	-1002	4:06

Table 3: Operators assemble on 50,100,150 and 200 numbers of generations with fix size of population and calculated fitness score, running time

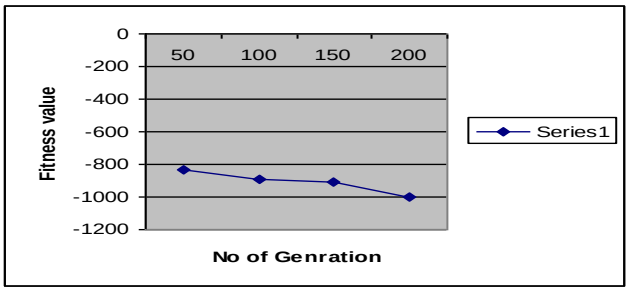

Figure9: Fitness curve GA with Verifying number of generation

In the proposed solution, we have also used two specific crossover and mutation operators. In order to determine the best crossover and mutation probabilities; we have carried out three different experiments, using ten randomly selected canis_familiaris dataset that were obtained from [15]. In our experiments for each of ten datasets, the algorithm is executed 200 run on GA and the statistical outcomes of the optimal fitness in each run is calculated as the results. We measure the best fitness score and running time for each generation. Our algorithm has to be run with the 30% crossover & 10%-mutation option, 60% crossover & 30% mutation option and 80% crossover & 50% mutation option .it is observed that our algorithm obtained the best solutions for 80% crossover & 50% mutation option .The solutions obtained by the 60% crossover and 30% mutation for the same datasets are close to the best scores, however the option 30% crossover & 10% mutation has not achieved any good quality solutions. Therefore, we can conclude that GA has achieved overall better performance for these test datasets when the rate of crossover are selected as 80% and mutation are selected as 50%. As for results for these datasets are to be presented in table 4 and corresponding plots are to be presented in graph 10.

Cros rate=30% Mut rate=10%	Cros rate=60% Mut rate=30%	Cros rate=80% Mut rate=50%	Pop Size	Running Time
-950	-940	-920	5	1:03
-935	-923	-900	10	2:05
-900	-868	-860	15	3:07

Table 4: fitness scores with selected Crossover and mutation rate options

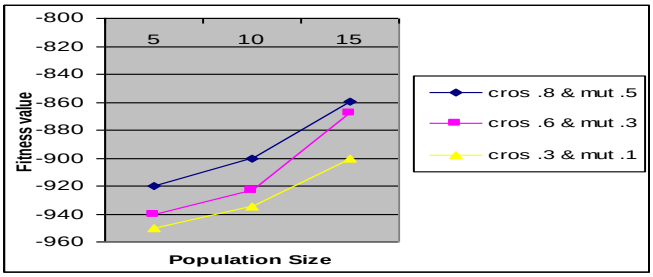

Figure10: Experimental results on GA with selected Crossover and mutation rate options

The last set of experiments compares our algorithm (GAMS) with two different tools such as Maft (high speed multiple sequence alignment program) and Kalign (fast and accurate multiple sequence alignment algorithm). The maximum 200 generation run on GA and the statistical outcomes of the optimal fitness in each run is calculated as the results. The sequence id and specification of each dataset is given in table 5. We measure that our GA obtained the best fitness score and running time for each generation as compare to other. The GA typically found a good alignment within 200 generations. Table 6 and graph 10 lists the results after comparisng our algorithm with Maft and kalign.

Seq ID	Sequence Specification	No of Sequence
A1	>SPAC1142.01 >SPAC1002.04c >SPAC105.03c >SPAC10F6.09c	4
A2	>SPAC1006.08 >SPAC1006.05c >SPAC1002.20 >SPAC10F6.08c	4
A3	>SPAC1002.02 >SPAC11E3.06 >SPAC1002.08c >SPAC10F6.03c	4
A4	>SPAC1002.13c >SPAC11D3.01c| >SPAC10F6.14c >SPAC1071.07c	4

Table 5: Canis_familiaris dataset

Seq ID	GAMS	Maft	Kalign
A1	-4679	-6446	-6795
A2	-2367	-3469	-3868
A3	-4317	-5814	-6557
A4	-3408	-5450	-6411

Table 6: Overall Performance of all methods of Sequence ID datasets

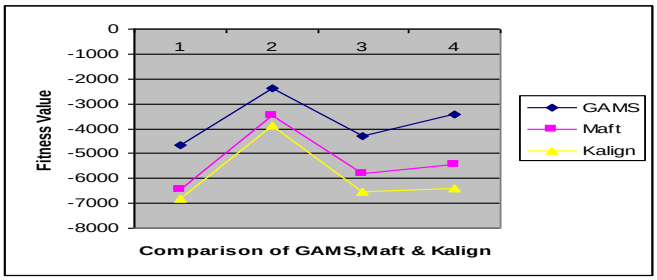

Figure10: Overall Performance of all methods of Sequence ID datasets

V. CONCLUSION

Multiple sequence alignmen is an extension of pairwise alignment to incorporate more than two sequences at a time. Our multiple alignment methods try to align all of the sequences in a given query set. Efficient fitness value function, crossover and mutation strategies are the outcome of work. Eventually it is trying that our methods will be significantly contributed in prior efficient solution to multiple sequence alignment problems.

REFERNECES

[1] C. Gondro and B.P. Kinghorn, "A simple genetic algorithm for multiple Sequence alignment", Genetics and Molecular Research 6 (4): 964-982 (2007)

[2] Kosmas Karadimitriou and Donald H. Kraft, "Genetic Algorithm and the Multiple Sequence Alignment in Biology ", Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, February 1996, Baton Rouge, LA.

[3] C. Notredame, « Recent progresses in MSA a survey. , pharmacogenomic, volume 3, pages 1–14, 2002.

[4] Fernando José Mateus da Silva, Juan Manuel Sánchez Pérez, Juan Antonio Gómez , "Optimizing Multiple Sequence Alignment by Improving Mutation Operators of a Genetic Algorithm", 978-0-7695-3872-3/09 © 2009 IEEE

[5] C. Notredame and D.G. Higgins. "SAGA: sequence alignment by genetic algorithm", Nucleic Acids Research, volume 24(8): 1515–1524, 1996.

[6] L. Davis, "Handbook of Genetic Algorithms." Van Nostrand Reinhold, New York, 1991.

[7] Zhang C, Wong, "AKC: Toward efficient multiple molecular sequence alignment: a system of genetic algorithm and dynamic programming", IEEE Transactions on Systems, Man and Cybernetics, Part B 1997, 27:918 -932.

[8] Nguyen HD, Yamamori K, Yoshihara I, Yasunaga M, "Improved GA-based method for multiple protein sequence alignment", The 2003 Congress on Evolutionary Computation (CEC '03) 2003, 3:1826 - 1832.

[9] Shyi-Ming Chen, Chung-Hui Lin, and Shi-Jay Chen, "Multiple DNA Sequence Alignment Based on Genetic Algorithms and Divide-and-Conquer Techniques", International Journal of Applied Science and Engineering (2005). 3, 2: 89-100

[10] Jorng-Tzong Horng, Li-Cheng Wu, Ching-Mei Lin, Bing-He Yang, "A Genetic Algorithm For Multiple Sequence Alignment", Soft Computing-A Fusion of Foundations, Methodologies and Applications, Vol. 9, Issue 6, pp 407 – 420. (2005)

[11] Yang Chen, Jinglu Hu, Member, IEEE, Kotaro Hirasawa, Member, IEEE, Songnian Yu. "Multiple Sequence Alignment Based on Genetic Algorithms with Reserve Selection" , ICNSC, pp 1511-1516 (2008).

[12] Fernando José Mateus Silva, Member, IEEE, Juan Manuel Sánchez-Pérez, Juan Antonio Gómez-Pulido and Miguel A. Vega-Rodríguez , "An Evolutionary Approach for Performing Multiple Sequence Alignment", 978-1-4244-8126-2/10/$26.00 ©2010 IEEE

[13] Amouda Nizam,Jeyakodi Ravi1, and Kuppuswami Subburaya2, "Cyclic Genetic Algorithm for Multiple Sequence Alignment", International Journal of Research and Reviews in Electrical and Computer Engineering (IJRRECE) Vol. 1, No. 2, June 2011

[14] Guang-Zheng Zhang De-Shuang Huang,"Aligning Multiple Protein Sequence by AnImproved Genetic Algorithm", IEEE 0-7803-8359-1/04/$20.00 0 2004

[15] Canis_familiaris.BAOADD2.66.pep.abinitio.fa.gz.