A Genetic Algorithm for Alignment of Multiple DNA Sequences

Pankaj Agarwal[1], Ruchi Gupta[2], Taru Maheswari[2], Princy Agarwal[3], Shubhanjali Yadav[3], and Vishnu Bali[3]

[1] Department of Computer Science & Engineering,
IMS Engineering College, Ghaziabad, U.P, India
pankaj7877@gmail.com
[2] Department of MCA, AKGEC, Ghaziabad, U.P, India
80ruchi@gmail.com
[3] Department of Computer Science & Engineering,
IMS Engineering College, Ghaziabad, India

Abstract. This paper presents a new genetic algorithm based solution to obtain alignment of multiple DNA molecular sequences. Multiple Sequence alignment is one of the most active ongoing research problems in the field of computational molecular biology. Sequence alignment is important because it allows scientists to analyze protein strands (such as DNA and RNA) and determine where there are overlaps. These overlaps can show commonalities in evolution and they also allow scientists to better prepare vaccines against viruses, which are made of protein strands. We have proposed new genetic operations for crossover, mutation, fitness calculation, population initialization. Proposed scheme generates new populations with better fitness value. We have also reviewed the some of the popular works by different researchers towards solving the MSA problem w.r.t various phases involved in general GA procedure. A working example is presented to validate the proposed scheme. Improvement in the overall population fitness is also calculated.

Keywords: Multiple Sequence Alignment, Genetic Algorithms, NP-Complete, Computational Biology etc.

1 Introduction and Related Work

Multiple sequence alignment (MSA) refers to the problem of optimally aligning three or more sequences of symbols with or without inserting gaps between the symbols. The objective is to maximize the number of matching symbols between the sequences and also use only minimum gap insertion, if gaps are permitted. Multiple Sequence Alignment (MSA) helps in detecting regions of significant sequence similarity from collections of primary molecular sequences of DNA or proteins. Genetic algorithms have been found to be quite suitable strategy for aligning multiple sequences. In this section, a review of the literature on how genetic algorithms have been used to solve the MSA problem will be given. Each phase of the genetic algorithm is reviewed and the similarities and differences will be noted.

V.V. Das and J. Stephen (Eds.): CNC 2012, LNICST 108, pp. 437–443, 2012.

Initial Populations: The first challenge of a genetic algorithm is to determine how the individuals of the population will be encoded and to generate an initial population with some degree of randomness. Literature review **[2,3,4]** suggests that each individual in the population should be one multiple alignment of all the given sequences, but the way that they came up with the initial population varies. In **[3,4,5]** the sequence length is increased by a certain percentage and randomly inserted gaps or buffers of gaps into the sequences were considered. Hernandez et. Al [2] took a new approach and used previously developed tools to align the sequences to a certain degree and then used the GA to optimize the alignment.

Reproduction: Researchers Hernandez et al.**[2]**, 2004; Horng et al.**[3]**, 2005; Shyu et al., 2006**[5]**; Wang & Lefkowitz**[4]**, used the typical tournament style, also known as, "roulette wheel" style of reproduction. Two of them **[3, 4]** also used some sort of elitism while further restrictions were made by Wang & Lefkowitz [5] to only allow the top scores to reproduce.

Crossover: After reproduction, pairs of alignments from the old population are randomly chosen for crossover. The most common type of crossover is called "One Point Crossover". Hernandez et al. **[2]** and Shyu et al.**[5]** used the process of dividing the sequences in the alignments at a random point, and then swapping the first halves of the first alignment with the first halves of the second alignment.

Mutation: Mutation is the last step in the process. There are a few ways to do mutation for this problem and they all have to do with gaps and sliding subsequences left or right. Hernandez et al. **[1]** have two forms of mutation. They either remove a gap or slide a sub-sequence next to a gap into the gap space which essentially moves the gap from the beginning of the sub-sequence to the end or vice versa. Homg et al. **[3]** have four forms of mutation. Shyu et al. **[5]** randomly select columns in the sequences and then swap nucleotides and spaces in these columns. Wang & Lefkowitz **[4]** have three forms of mutation.

Fitness Function: The fitness function determines how "good" an alignment is. The most common strategy that is used by all of the authors, albeit with significant variations, is called the "Sum-Of-Pair"

Hernandez et al.**[1]** ***and*** Wang & Lefkowitz **[4]** create their own scoring matrices based upon the sequences that they are trying to align. Wang & Lefkowitz **[4]** creates a library of optimal pair-wise alignments from the sequences that they are trying to align and then evaluates the consistency of the calculated multiple alignment with the ones in the library. Homg et al. **[3]** uses the straight-forward Sum-of-Pairs calculation. Shyu et al. **[5]** uses the nucleic acid scoring matrix from the International Union of Biochemistry (IUB). This matrix groups nucleotides together according to certain properties, e.g., Purines (A or G) and Pyrimidines (C or T). Segun et. al **[8]** describes that MSA belongs to a class of optimization problem called the combinatorial problems with exponential time complexity $O(L^N)$. Literature review **[9, 10]** describe how Genetic Algorithms can be used to solve the MSA problem and that optimal, or near-optimal solutions using GA Kosmas et. al **[10]** showed GA to be better than other optimization methods as it require only a fitness function, rather no particular algorithm to solve a given problem. The fitness function is the cost function given using different weights for different types of matching symbols and assigning gap costs **[11]**. Nizam et. Al **[12]** proposed a scheme where parameters like number of

generations, chromosome length, crossover and mutation rate are made to adapt the values during execution using concept of self-organizing GA. Yang Chen et. al **[13]** is based on Genetic Algorithm with Reverse Selection(GARS).One of the drawback with Genetic Algorithm is that it suffers from premature convergence in which solution reaches locally optimal stage.

2 Proposed Genetic Algorithm

2.1 Pseudo Code

1. **Initialization:** Calculate a length of sequences in the alignment by stating the maximum number of gaps permitted in the maximum length sequence given, let say this length is given by ***length***. Now, Generate the initial alignment of n sequences by inserting the required number of gaps, ***length-Sequence_Length(i).***
2. **Chromosome representation:** Using the given representation Scheme, encode sequence alignment into chromosomes. Repeating, steps 2 and 3 generate any desired number of chromosomes of given sequences, required to perform various operations.
3. **Fitness Evaluation:** Evaluate the fitness f(x) of each chromosome that is showing the score for its respective sequence alignment.
4. **Selection:** Select and save the elite (chromosome with highest fitness value) in the current population. Perform Operations on the remaining chromosomes.
5. **Genetic Operations:** Create a new population using (a to f) repeatedly until the desired fitness threshold value is not obtained:
    a. Save the best of chromosomes in the current population.
    b. Perform ***crossover*** operations on the pairs of less fit chromosomes parent[i] and parent[i+1].
    c. Discard the chromosomes produced whose fitness value is less than the parent chromosomes. i.e. the best 2 chromosomes of parent[i], parent[i+1], child[i], child[i+1] .
    d. Find the mutation point based on the specified optimal rate for all the selected sequences.
    e. Perform ***mutation***. Calculate overall alignment fitness value of alignment.
    f. If fitness value of old population is higher, discard changes done in mutation.
    g. Save the chromosome alignment representation.
6. The best Sequence Alignment would be corresponding to the chromosome with highest fitness value from among all chromosomes of the last generation.

2.2 Example Demonstration

Let us consider the following sample of four sequences set:
AAAGCTAT
GATACAA
ACCTTAAA

ATAGAAGGT

The initial Population is the chromosome representation of various multiple sequence alignments generated by randomly inserting gaps in the input Sequences.

(i) A-AAGCT--AT
GA-T-A-C-AA
A-CC-TTAA-A
A-TAGAAGG-T
Fitness= -52 1 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11

Similarly ten other chromosomes are generated by inserting the gaps randomly to form the initial population:

(ii) Fitness= -73 6 7 8 11 0 2 6 9 11 1 7 8 11 3 5 11
(iii) Fitness= -35 2 4 9 11 0 3 7 9 11 0 1 9 11 2 4 11
(iv) Fitness= -43 1 5 7 11 0 4 7 8 11 0 2 4 11 4 5 11
(v) Fitness= -54 1 5 8 11 0 1 3 8 11 0 6 8 11 5 7 11
(vi) Fitness= -53 1 2 5 11 1 3 4 5 11 1 2 9 11 0 2 11
(vii) Fitness= -35 2 3 9 11 2 5 6 9 11 0 2 5 11 0 3 11
(viii) Fitness= -77 0 4 7 11 2 5 6 8 11 1 2 9 11 4 5 11
(ix) Fitness= -54 1 4 7 11 0 1 6 9 11 0 5 9 11 0 4 11
(x) Fitness= -45 3 5 8 11 4 6 7 8 11 3 5 8 11 1 7 1

For the given Input Sequence: Overall Fitness Score= -521

Selection: Save 20% and perform operations on the remaining 80%. For the 10 chromosomes of Initial generation, save **(iii)** and **(vii)** which pass to next generation without undergoing any change. Perform genetic operations on the remaining eight chromosomes. Input for Crossover are given as:

1 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -52 parent 1
6 7 8 11 0 2 6 9 11 1 7 8 11 3 5 11 Fitness= -73 parent 2
1 5 7 11 0 4 7 8 11 0 2 4 11 4 5 11 Fitness= -43 parent 3
1 5 8 11 0 1 3 8 11 0 6 8 11 5 7 11 Fitness= -54 parent 4
1 2 5 11 1 3 4 5 11 1 2 9 11 0 2 11 Fitness= -53 parent 5
0 4 7 11 2 5 6 8 11 1 2 9 11 4 5 11 Fitness= -77 parent 6
1 4 7 11 0 1 6 9 11 0 5 9 11 0 4 11 Fitness= -54 parent 7
3 5 8 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -45 parent 8

After *crossover*, the new chromosomes generated would be:

1 7 8 11 0 2 6 9 11 1 7 8 11 3 5 11 Fitness= -67 Child 1
6 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -60 Child 2
1 5 7 11 0 1 3 8 11 0 6 8 11 5 7 11 Fitness= -59 Child 3
1 5 8 11 0 4 7 8 11 0 2 4 11 4 5 11 Fitness= -40 Child 4
1 2 5 11 2 5 6 8 11 1 2 9 11 4 5 11 Fitness= -40 Child 5
0 4 7 11 1 3 4 5 11 1 2 9 11 0 2 11 Fitness= -66 Child 6
1 4 7 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -49 Child 7
3 5 8 11 0 1 6 9 11 0 5 9 11 0 4 11 Fitness= -66 Child 8

Selection: Best 2 chromosomes Of parent[i], parent[i+1], child[i], child[i+1] are passed to next step for i=1,3,5,7

1 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -52 Parent 1

6 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -60 Child 2
1 5 7 11 0 4 7 8 11 0 2 4 11 4 5 11 Fitness= -43 Parent 3
1 5 8 11 0 4 7 8 11 0 2 4 11 4 5 11 Fitness= -40 Child 4
1 2 5 11 2 5 6 8 11 1 2 9 11 4 5 11 Fitness= -40 Child 5
1 2 5 11 1 3 4 5 11 1 2 9 11 0 2 11 Fitness= -53 Parent 5
3 5 8 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -45 Parent 8
1 4 7 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -49 Child 7

These would be the Input for the next step i.e. Mutation.

Before And After mutation, the chromosomes are:

2 4 9 11 0 3 7 9 11 0 1 9 11 2 4 11 Fitness= -35 **Best Fitness value Chromosome Saved**
2 3 9 11 2 5 6 9 11 0 2 5 11 0 3 11 Fitness= -35 **Best Fitness value Chromosome Saved**
1 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -52 **Kept in the SELECTION followed**
1 7 8 11 2 5 6 8 11 1 4 9 11 1 9 11 Fitness= -57 **Discarded**
6 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -60 **Kept**
6 7 8 11 2 4 6 8 11 1 4 9 11 1 10 11 Fitness= -69 **Discarded**
1 5 7 11 0 4 7 8 11 0 2 4 11 4 5 11 Fitness= -43 **Discarded**
1 5 7 11 0 4 7 8 11 1 2 4 11 4 5 11 Fitness= -41 **Kept**
1 5 8 11 0 4 7 8 11 0 2 4 11 4 5 11 Fitness= -40 **Kept**
1 5 8 11 0 4 9 8 11 0 2 4 11 4 5 11 Fitness= -40 **Discarded**
1 2 5 11 2 5 6 8 11 1 2 9 11 4 5 11 Fitness= -40 **Kept**
1 2 5 11 2 5 6 8 11 1 2 9 11 4 7 11 Fitness= -49 **Discarded**
1 2 5 11 1 3 4 5 11 1 2 9 11 0 2 11 Fitness= -53 **Discarded**
1 2 5 11 1 6 4 5 11 1 2 9 11 0 2 11 Fitness= -38 **Kept**
3 5 8 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -45 **Kept**
3 5 8 11 4 6 9 8 11 3 5 8 11 1 7 11 Fitness= -52 **Discarded**
1 4 7 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -49 **Kept**
1 4 7 11 4 6 9 8 11 3 5 8 11 1 7 11 Fitness= -63 **Discarded**

Selection- Changes done in mutation are discarded if fitness doesn't increase.

2 4 9 11 0 3 7 9 11 0 1 9 11 2 4 11 Fitness= -35
2 3 9 11 2 5 6 9 11 0 2 5 11 0 3 11 Fitness= -35
1 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -52
6 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 Fitness= -60
1 5 7 11 0 4 7 8 11 1 2 4 11 4 5 11 Fitness= -41
1 5 8 11 0 4 7 8 11 0 2 4 11 4 5 11 Fitness= -40
1 2 5 11 2 5 6 8 11 1 2 9 11 4 5 11 Fitness= -40
1 2 5 11 1 6 4 5 11 1 2 9 11 0 2 11 Fitness= -38
3 5 8 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -45
1 4 7 11 4 6 7 8 11 3 5 8 11 1 7 11 Fitness= -49

After One Iteration of Selection, Crossover And Mutation the chromosomes are:

2 4 9 11 0 3 7 9 11 0 1 9 11 2 4 11 **Fitness= -35**
2 3 9 11 2 5 6 9 11 0 2 5 11 0 3 11 **Fitness= -35**
1 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 **Fitness= -52**
6 7 8 11 2 4 6 8 11 1 4 9 11 1 9 11 **Fitness= -60**

1 5 7 11 0 4 7 8 11 1 2 4 11 4 5 11 **Fitness= -41**
1 5 8 11 0 4 7 8 11 0 2 4 11 4 5 11 **Fitness= -40**
1 2 5 11 2 5 6 8 11 1 2 9 11 4 5 11 **Fitness= -40**
1 2 5 11 1 6 4 5 11 1 2 9 11 0 2 11 **Fitness= -38**
3 5 8 11 4 6 7 8 11 3 5 8 11 1 7 11 **Fitness= -45**
1 4 7 11 4 6 7 8 11 3 5 8 11 1 7 11 **Fitness= -49**

Next Generation Produced, Output Sequence: Overall Fitness Score= **-435**

The Best Alignment is defined by the chromosome with highest fitness value:

2 3 9 11 2 5 6 9 11 0 2 5 11 0 3 11 Fitness= -35

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

3 Results

We've used the defined operators in our implementation for 4 sequences that initially generated 10 chromosomes. Various operators developed by us were then applied on these 10 chromosomes an iteration and it was observed that the overall fitness score for next generation (population) of chromosomes was better than the previous one. Success rate computated in terms of % is quite significant and the proposed scheme can be used to generate good multiple alignments.

Table 1. Operators used in Implementation for 4 sequences

No. of sequences	Size Of Population	CrossOver Rate (R_c)	Mutation Rate (R_m)	Fitness 1st Gen.	Fitness 2nd Gen.	Scoring Matrix	GAP Penalty
4	10	0.6	0.3	-521	-435	PAM250	-3

Table 2. Operators deployed on 10 chromosomes And Calculated Success Rate

No. Of Chromosomes	**Population Size**	**Fitness Sc. 1st Gen.**	**Fitness Sc. 2nd Gen.**	**Fitness Sc. 3rd Gen.**	**Success Rate**
4	10	-521	-435	-410	11.10%
6	12	-639	-505	-460	10.49%
10	16	-650	-520	-480	13.80%
12	20	-665	-535	-485	14.25%

References

1. Goldberg, D.E.: Genetic algorithms in search, optimization & machine learning. Addison-Wesley Publishing Company, Inc., Reading (1989)
2. Hernandez, D., Grass, R., Appel, R.: MoDEL: an efficient strategy for ungapped local multiple alignment. Computational Biology and Chemistry 28, 119–128 (2004)
3. Horng, J.T., Wu, L.C., Lin, C.M., Yang, B.H.: A genetic algorithm for multiple sequence alignment. Soft Computing 9, 407–420 (2005)
4. Wang, C., Lefkowitz, E.J.: Genomic multiple sequence alignments: Refinement using a genetic algorithm. BMC Bioinformatics 6, 200 (2005)
5. Shyu, C., Sheneman, L., Foster, J.A.: Multiple sequence alignment with evolutionary computation. Genetic Programming and Evolvable Machines 5, 121–144 (2004)
6. Buscema, M.: Genetic doping algorithm (GenD): Theory and applications. Expert Systems 21(2), 63–79 (2004)
7. Notredame, C., Higgins, D.G.: SAGA: Sequence alignment by genetic algorithm. Nucleic Acids Research 24(8), 1515–1524 (2004)
8. Fatumo, S.A., Akinyemi, I.O., Adebiyi, E.F.: Aligning Multiple Sequences with Genetic Algorithm. International Journal of Computer Theory and Engineering 1(2), 186–190 (2009)
9. Carrillo, H., Lipman, D.: The multiple sequence alignment problem in biology. Siam J. Appl. Math. 48(5), 1073–1082 (1988)
10. Kosmas, K., Donald, H.K.: Genetic Algorithms and the Multiple Sequence Alignment Problem in Biology. In: Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, Baton Ronge, LA (February 1996)
11. Altschul, S.F.J.: Gap Costs for Multiple Seqence Alignment. Heoretical Biol. 138, 297–309 (1989)
12. Nizam, A., Shanmugham, B., Subburaya, K.: Self-Organizing Genetic Algorithm for Multiple Sequence Alignment (2010)
13. Chen, Y., Hu, J., Hirasawa, K., Yu, S.: Multiple Sequence Alignment Based on Genetic Algorithms with Reserve Selection. In: ICNSC, pp. 1511–1516 (2008)