International Association of Scientific Innovation and Research (IASIR)
(An Association Unifying the Sciences, Engineering, and Applied Research)

ISSN (Print): 2279-0047
ISSN (Online): 2279-0055

International Journal of Emerging Technologies in Computational and Applied Sciences(IJETCAS)

www.iasir.net

Alignment of Multiple Sequences using GA method

Dr. Pankaj Agarwal
Professor, Department of Computer Science & Engineering, IMS Engineering College, Ghaziabad, U.P, India-201009,
Rahul Chauhan
Azad Institute 0f Engineering & Techology, Lucknow
Princy Agarwal, Taru Maheswari, Shubhanjali Yadav, Vishnu Bali
B.Tech (2008-12 batch), Department of Computer Science & Engineering,
IMS Engineering College, Ghaziabad, India

Abstract: *This paper proposes few genetic operators to obtain better alignments of multiple molecular sequences. Datasets from DNA families of Canis_familiaris dataset have been considered for experimental work & analysis. All the proposed operators in the method have been implemented and validated within a self developed software tool which allows the user to select the various genetic operators for crossover, mutation, fitness calculation, population initialization. It guarantees the next generation of populations with better fitness value. Improvement in the overall population fitness is also calculated and evaluated. Survival of the fittest policy is followed to arrive at a better fitness in following generations. Observations based on variable parameters have been recorded, analyzed & presented in the form of results. Results were also compared with few standard existing online tools to study the feasibility of the proposed operators.*

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

I. Introduction

Multiple Sequence Alignment (MSA) is one of the most challenging and active ongoing research problems in the field of computational molecular biology. Multiple sequence alignment of DNA, RNA, or amino acids is essential for biologists to study similarity in sequences which often leads to similarity in function and provides valuable evolutionary information. The alignment enables us to infer the evolutionary history of the sequences.

Because three or more sequences of biologically relevant length can be computationally quite difficult. Efficient computational algorithms can be used to produce better results. The basic problem with MSA algorithms is their exponential complexity with the considerably large input data set. Infact computationally MSA is considered to NP Complete problem and therefore most multiple sequence alignment programs use heuristic methods **[1]**.

Proposed algorithm always guarantees that the next obtained generation of populations will have better fitness value as compared to their ancestors and therefore we can expect that the tool provides at least near to optimal alignments after some N number of generations. During the generation of the next population the tool ensures that only the fitter candidates (here alignments) are considered and weaker ones are ignored. The overall purpose remains to improve the alignment with each generation.
The proposed tool offers some of the advantages:
1. It always guarantees that the next obtained generation of populations will have better fitness value
2. Facility given to the users to set various GA parameters like crossover rate, mutation rate, no. of generations, selection of various implemented GA operations like selection, crossover or mutation schemes.

II. Proposed Genetic Algorithm

The pseudo code is as follows:

1. **Start**
2. **Initialization: S**equence length is computed after finding maximum number of gaps allowed with respect to the longest sequence in the set of sequences that needs to be aligned. Let say the aligned sequences' length is given by ***length***, generate initial alignment by inserting required number of gaps given by, ***length-***

sequence_Length(i). An initial population of several alignments is created in this manner. Size of the initial population can be set by the user as well.

3. **Chromosome Representation:** Encode the alignments of initial population into chromosomes using the representation scheme described later in the section**.**
4. **Genetic Operations:** Create a new population using following steps repeatedly, until the minimum desired fitness value is not obtained or desired N generations are done :
   - ***Selection:*** Using selection schemes like **elitism** or **random selection**, few sequences are selected to perform crossover & mutation operations.
   - ***Crossover*** operations are performed on the pairs of less fit chromosomes. **Single point** crossover, **double point** crossover and **min-max** crossover methods have been used.
   - ***Selection for next generation:*** chromosomes with better fitness values among the lot are used for producing other fit chromosomes using crossover and mutation schemes. Here we have experimented with a simple scheme where the chromosomes produced whose fitness value is less than the parent chromosomes are discarded. i.e. the best 2 chromosomes of $parent_i$, $parent_j$, $child_i$ and $child_j$. One & two point crossover schemes are tried.
   - ***Mutation*** operation is performed on selected chromosomes**.** Following mutations are performed- **random gap shuffling**, **insertion and deletion** of gaps.
   - Calculate overall alignment fitness value of the obtained alignments from crossover & mutation operations.
   - Discard the chromosomes, whose fitness value is less than the parent chromosomes.

   Save the alignment representation & its associated parameters.

5. **Result:** The best sequence alignment would be corresponding to the chromosome with highest fitness value after N generations are done or desired minimum acceptable score is obtained.
6. **End**

Example Demonstration: INPUT SEQUENCE

>MMVHLTPMMKSAVTALWGKVNVDMVGGMALGRLLVVYPWTQRFFMSFGDLSTPDAVM
>MMGLSDGMWQLVLNVWGKVMADIPGHGQMVLIRLFKGHPMTLMKFDKFKHLKSMDMMKAS
>ALVMDNNAVAVSFSMMQMALVLKSWAILKKDSANIALRFFLKIFMVAPS
>MMRPMPMLIRQSWRAVSRSPLMHGTVLFARLFALMPDLLPLFQYNCRQFSSPMD

A. Initialization

We insert gaps in the input sequence to make the initial population of say 10 alignments.

Pseudocode: init_pop (*lmax:* length of longest input sequence, len_i: length of ith input sequence, iPop: initial population set)

```
{
    Calculate the length of sequence in alignment using length (N)= 1.2* lmax
    For k= 1 to ipop
    {
        For each sequence i=1 to N
            compute no. of gaps to be inserted as $gap_i$= length-$len_i$.
        For each sequence i=1 to N
            insert $gap_i$ number of gaps at random positions.
    this initial alignment is referred using $seq\text{-}al_k$.
}
      }
```

lmax = 61, corresponding to the longest sequence, ***length(N)***=1.2 * ***lmax*** **=74, gap_1= *length*-len_1 =17, gap_2= *length*-len_2 =13, gap_3= *length*-len_3 =25, gap_4= length-len_4 =20**

Initial Population's Single Alignment Instance After insertion of gaps, as stated in the algorithm:

>MMVHLT---PM---MKSAV-T-AL-WGKVNVDMVGGMALGR--LLV-VYPWTQ-R-FFMSF-GDLSTPDA--VM
>M-MGL--SDGM-WQ-LVL-N--VW-GKVM-ADIP-GHGQMVLIRLFKGHPMTL-MKFDKF-KHLKSMD-MMKAS
>A-LVMDNNA--VAV--S--FS--MM-Q--MA--LVL-KS-W-A-ILKKD---S-A-N-IALRFFLKIFM-VAPS
>MMRP-MPML-I-RQSWR--AVS-RS-P-LMHGT-VLF-ARLFALM--PDLLP--L---FQ-YNCRQF-SSP-MD

B. Chromosome Representation

The chromosomes are generated by encoding the sequences. The gap positions in the sequence are being used to represent a chromosome. The gap positions of all the sequences are used to make a single chromosome where, end of a sequence in chromosome representation is indicated by a complete point. A complete point's value is equal to the length of the each sequence in the initial population. In this manner ten chromosomes are produced corresponding to initial population of 10 alignments.

```
Pseudocode: chrom_rep (seq-al_k: initial population, length: length of each sequence in alignment) {
        For k=1 to iPop   // corresponding to each alignment of initial population
        {
            For each sequence i=1 to N                // length(N)=1.2 * lmax
                For each gap in sequence i from seq- al_k,
                        Insert position of each gap, in the chromosome chrom_k.
                Insert length in chrom_k denoting the complete point.
            chrom_k is the final chromosome representation for alignment, seq-al_k.
        }
}
```

For the above Alignment, the chromosome is given by representing the position of each gap in sequences:

Chromosome Representation:

***chrom*₁** 6 7 8 11 12 13 19 21 24 41 42 46 53 55 61 70 71 74 1 5 6 11 14 18 20 21 24 29 34 53 60 74 1 9 10 14 15 17 18 21 22 25 27 28 31 32 36 39 41 43 49 50 51 53 55 57 69 74 4 9 11 17 18 22 25 27 33 37 45 46 52 53 55 56 57 60 67 71 74

Similarly nine other chromosomes are generated by inserting the gaps randomly to form the initial population.

***chrom*₂** 0 2 4 7 8 14 17 21 23 24 28 30 34 37 57 59 71 74 4 15 22 26 28 31 32 35 39 41 53 70 71 74 2 3 6 13 17 18 21 22 28 29 33 38 39 45 46 48 51 57 58 59 64 65 66 69 70 74 0 2 7 8 11 24 26 30 31 35 39 40 41 43 44 46 49 57 68 69 74

***chrom*₃** 0 1 2 10 12 17 18 19 20 32 36 38 39 49 50 52 61 74 1 7 8 9 10 17 19 29 36 40 61 67 70 74 1 6 9 12 13 14 15 16 17 18 21 23 27 28 31 33 35 44 45 46 48 65 66 71 72 74 0 1 5 6 9 10 17 18 19 20 26 27 30 31 35 36 42 46 57 64 74

***chrom*₄** 1 2 3 4 8 17 20 36 37 38 40 43 47 52 54 55 64 74 1 6 10 11 24 33 34 42 47 54 56 69 71 74 1 7 8 10 11 12 14 17 18 19 20 21 22 28 32 33 36 37 39 46 49 53 57 62 68 74 1 2 11 13 16 21 22 23 25 26 29 39 41 43 45 50 56 57 65 69 74

***chrom*₅** 0 6 8 9 11 12 16 18 19 22 29 30 32 33 34 49 63 74 0 3 12 14 15 20 21 24 42 46 51 54 69 74 0 2 5 6 8 12 13 17 18 22 29 31 35 39 40 41 44 45 47 49 50 56 64 65 66 74 0 1 3 5 6 9 10 11 14 18 19 23 24 32 34 37 45 52 53 60 74

***chrom*₆** 4 8 11 18 20 25 27 28 33 41 46 51 52 57 59 64 66 74 5 8 17 18 23 30 37 38 41 44 45 62 72 74 5 7 14 15 16 18 20 21 23 25 26 27 28 32 33 40 41 43 49 50 55 59 64 67 71 74 0 2 4 5 8 17 30 33 37 39 41 43 45 47 50 51 52 55 62 68 74

***chrom*₇** 0 6 17 19 23 24 26 31 32 38 46 51 52 60 64 67 68 74 1 2 5 11 26 35 37 41 48 54 57 58 64 74 0 6 7 8 12 14 20 21 23 25 33 34 35 36 41 44 48 50 55 58 61 62 63 70 72 74 0 4 8 9 10 16 17 18 24 26 27 39 42 50 51 55 56 60 63 71 74

chrom$_8$ 0 4 6 11 16 17 27 28 30 33 38 47 49 62 65 68 71 74 0 4 12 14 15 16 20 26 28 29 31 45 56 74 0 2 3 4 8 9 10 11 14 19 21 24 25 26 27 28 32 33 34 37 40 47 49 51 55 74 0 2 5 7 9 13 14 23 24 28 33 40 46 50 51 55 62 67 69 70 74

chrom$_9$ 3 9 10 13 17 18 19 21 25 30 32 38 46 50 53 54 64 74 3 4 12 15 20 22 28 30 32 40 44 47 60 74 3 6 7 8 9 10 16 19 24 27 28 33 34 35 36 37 39 41 43 45 46 55 62 68 70 74 1 2 5 15 16 19 25 26 29 32 34 36 38 39 42 44 45 57 58 65 74

chrom$_{10}$ 1 4 8 9 12 14 17 23 28 29 32 38 53 56 57 61 71 74 1 3 5 6 16 20 28 34 35 52 56 58 70 74 1 2 3 4 11 14 17 18 20 22 24 28 30 33 34 37 39 41 43 46 50 51 59 69 71 74 1 7 8 9 11 13 19 23 24 25 28 29 31 37 39 40 41 52 64 68 74

C. Reproduction/ Selection:

Reproduction is usually the first operator applied on population. Chromosomes are selected from the population to be parents to crossover and produce offspring. According to Darwin's Theory of survival of the fittest, the best ones should survive and create new offspring **[4]**. That is why reproduction operator is sometimes known as the selection operator. The various selection schemes that we used in our tool are:

C1. Elitism

In this method, first the best 20% chromosomes are copied to a new population. The rest chromosomes undergo genetic operations in a classical manner. Elitism can very rapidly increase the performance of GA because it prevents loosing the best-found solutions. The pseudo code of the Elitism Selection Scheme is as follows:

```
Pseudocode: Elitism (chrom-pop_m: chromosome generation of m chromosomes)
{
    For k=1 to m, corresponding to each chromosome of a population generation
    Calculate the fitness of chrom_k.
    For k=1 to m,
        Obtain the highest fitness values chromosomes from chrom-pop_m.
    Save the best chromosome to be part of next generation.
    Perform crossover, mutation operations on the remaining chromosomes.
    }
```

chrom$_1$ Fitness= -597	***chrom***$_2$ Fitness= -616	***chrom***$_3$ Fitness= -622
chrom$_4$ Fitness= -637	***chrom***$_5$ Fitness= -660	***chrom***$_6$ Fitness= -497
chrom$_7$ Fitness= -694	***chrom***$_8$ Fitness= -670	***chrom***$_9$ Fitness= -654
chrom$_{10}$ Fitness= -616		

Applying Elitism- the highest fitness value chromosome, is part of the next generation:
chrom$_6$ Fitness= -497

4 8 11 18 20 25 27 28 33 41 46 51 52 57 59 64 66 74 5 8 17 18 23 30 37 38 41 44 45 62 72 74 5 7 14 15 16 18 20 21 23 25 26 27 28 32 33 40 41 43 49 50 55 59 64 67 71 74 0 2 4 5 8 17 30 33 37 39 41 43 45 47 50 51 52 55 62 68 74

C2. Random Selection

In this method, any random chromosomes are copied to a new population. The rest chromosomes undergo genetic operations to produce new chromosomes. The pseudo code of the Random Selection Scheme is as follows:

```
Pseudocode: Random-Sel (chrom-pop_m: chromosome generation of m chromosomes)
{
    For k=1 to m,
        Choose any random number r and calculated i=(r%k)+1, which represents the chromosome to be
        selected
    Save the chrom_i to be part of next generation.
    Perform crossover, mutation operations on the remaining chromosomes.
    }
```

chrom$_1$ Fitness= -597	***chrom***$_2$ Fitness= -616	***chrom***$_3$ Fitness= -622

chrom$_4$ Fitness= -637 ***chrom***$_5$ Fitness= -660 ***chrom***$_6$ Fitness= -497
chrom$_7$ Fitness= -694 ***chrom***$_8$ Fitness= -670 ***chrom***$_9$ Fitness= -654
chrom$_{10}$ Fitness= -616

Applying selection- the chromosome which is part of the next generation:

chrom$_7$ Fitness= -694 0 6 17 19 23 24 26 31 32 38 46 51 52 60 64 67 68 74 1 2 5 11 26 35 37 41 48 54 57 58 64 74 0 6 7 8 12 14 20 21 23 25 33 34 35 36 41 44 48 50 55 58 61 62 63 70 72 74 0 4 8 9 10 16 17 18 24 26 27 39 42 50 51 55 56 60 63 71 74

D. **Crossover**

Cross over is a process of taking more than one parent chromosomes and producing a child solution from them. In our tool, we have implemented three types of crossovers- single point crossover, two point crossover and max-min crossover.

Cross over is performed by selecting two parents with higher fitness values as shown in example and then selecting a single crossover point which may be some formula based or randomly determined based on the length of the parents. Each such crossover results in two child chromosomes. As an experimental scheme we have restored only those child chromosomes which have better fitness scores than their parents.

For example consider the two parent chromosomes:

Parent$_1$ 0 2 4 7 8 14 17 21 23 24 28 30 34 37 57 59 71 74 4 15 22 26 28 31 32 35 39 41 53 70 71 74 2 3 6 13 17 18 21 22 28 29 33 38 39 45 46 48 51 57 58 59 64 65 66 69 70 74 0 2 7 8 11 24 26 30 31 35 39 40 41 43 44 46 49 57 68 69 74

Parent$_2$ 0 1 2 10 12 17 18 19 20 32 36 38 39 49 50 52 61 74 1 7 8 9 10 17 19 29 36 40 61 67 70 74 1 6 9 12 13 14 15 16 17 18 21 23 27 28 31 33 35 44 45 46 48 65 66 71 72 74 0 1 5 6 9 10 17 18 19 20 26 27 30 31 35 36 42 46 57 64 74

As an example cross over point can be calculated as:

Crossover point = 0.6*74 = 44, nearest complete point is 32nd position at which crossover performed.

However cross over point can be selected using other schemes as well

Child$_1$ 0 2 4 7 8 14 17 21 23 24 28 30 34 37 57 59 71 74 4 15 22 26 28 31 32 35 39 41 53 70 71 74 1 6 9 12 13 14 15 16 17 18 21 23 27 28 31 33 35 44 45 46 48 65 66 71 72 74 0 1 5 6 9 10 17 18 19 20 26 27 30 31 35 36 42 46 57 64 74

Child$_2$ 0 1 2 10 12 17 18 19 20 32 36 38 39 49 50 52 61 74 1 7 8 9 10 17 19 29 36 40 61 67 70 74 2 3 6 13 17 18 21 22 28 29 33 38 39 45 46 48 51 57 58 59 64 65 66 69 70 74 0 2 7 8 11 24 26 30 31 35 39 40 41 43 44 46 49 57 68 69 74

Performing min-max crossover on the parent chromosomes of above illustration:

Max-Child 0 2 4 10 12 17 18 21 23 32 36 38 39 49 57 59 71 74 4 15 22 26 28 31 32 35 39 41 61 70 71 742 6 9 13 17 18 21 22 28 29 33 38 39 45 46 48 51 57 58 59 64 65 66 71 72 74 0 2 7 8 11 24 26 30 31 35 39 40 41 43 44 46 49 57 68 69 74

Min-Child 0 1 2 7 8 14 17 19 20 24 28 30 34 37 50 52 61 74 1 5 8 9 10 17 19 29 36 40 53 67 70 741 3 6 12 13 14 15 16 17 18 21 23 27 28 31 33 35 44 45 46 48 65 66 69 70 74 0 1 5 6 9 10 17 18 19 20 26 27 30 31 35 36 42 46 57 64 74

E. **Mutation**

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of algorithm chromosomes to the next. Mutation alters one or more gene values in a chromosome from its initial state. After crossover the best set of chromosomes with number of chromosomes equaling to the size of **iPop** (initial population set) are selected and mutation is applied upon them.

Gap Shuffling

```
**Gap-Shuffle-Mut** (chrom-pop$_m$: chromosome generation of m chromosomes, mutP: mutation probability)
{
        for k=1 to m representing each chromosome, perform operations on decoded alignment sequence, seq_al$_k$ {
            for j=0 to mutP *length(seq) {
            for all sequences seq, find any gap in seq and shuffle it randomly within the seq.
            }
        Calculate the fitness of the new alignment.
            Out of parent and child, best alignment's chromosome becomes part of next generation.
    }
}
```

Illustration: Mutation

Before Mutation:

M--MVHLTPMMK-SAVTALWGKVNVDMVG-GMALGR-L-LV--VYP-WTQRFFMS-F-GDLSTPDAVM-GN

After Mutation:

MM-VHLT-PMMKSAV-TALWG-KVNVDMV-GGMALGRLL--VVYP-W-QRFFMSFG---DLSTPDAVMG-N

F. Fitness Function

The fitness function determines how "good" an alignment is. Fitness evaluation methods play an important role in the performance of evolutionary algorithms. The most common strategy that is used, albeit with significant variations, is called the "Sum-Of-Pair" Objective Function. In this method, for each location on the aligned sequences, one of three situations will occur: match, mismatch or a gap. The fitness of an alignment is calculated as **fitness = symReward – Pen(d,g)** where symReward is the overall reward of all pairwise symbol matches **[3]**. During the fitness evaluation, all fully-gapped columns in an alignment are ignored. For the Alignment as shown below,

s1	A	T	-	G	A	T	-	C	C	G
s2	-	T	A	G	C	T	A	C	C	-
s3	A	-	A	-	A	T	A	G	C	G

Fitness Score= fitness(s1,s2)+ fitness(s1,s3)+ fitness(s3,s2)

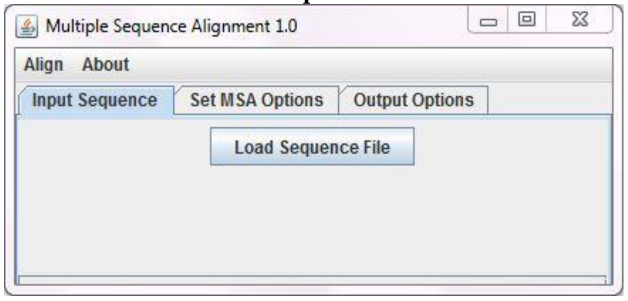
$$\text{Fitness value for n sequences} = \sum_{i=1}^{length} (\text{Fitness Score For Each pair of elements in the column})$$

Two scoring matrices have been used in our tool: **PAM-250** and, **BLOSUM-45**

Example: For The alignment given below and using PAM-250 Matrix and a gap penalty of -3, the overall fitness Score is calculated using above technique:

>MMVHLT---PM---MKSAV-T-AL-WGKVNVDMVGGMALGR--LLV-VYPWTQ-R-FFMSF-GDLSTPDA--VM
>M-MGL--SDGM-WQ-LVL-N--VW-GKVM-ADIP-GHGQMVLIRLFKGHPMTL-MKFDKF-KHLKSMD-MMKAS
>A-LVMDNNA--VAV--S--FS--MM-Q--MA--LVL-KS-W-A-ILKKD---S-A-N-IALRFFLKIFM-VAPS
>MMRP-MPML-I-RQSWR--AVS-RS-P-LMHGT-VLF-ARLFALM--PDLLP--L---FQ-YNCRQF-SSP-MD
Alignment Fitness Score= 547

III. Snapshots of Tool



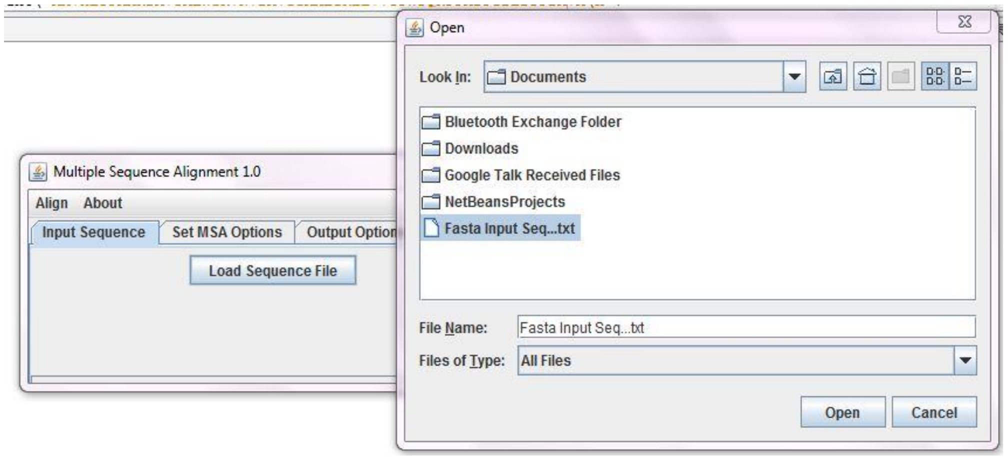
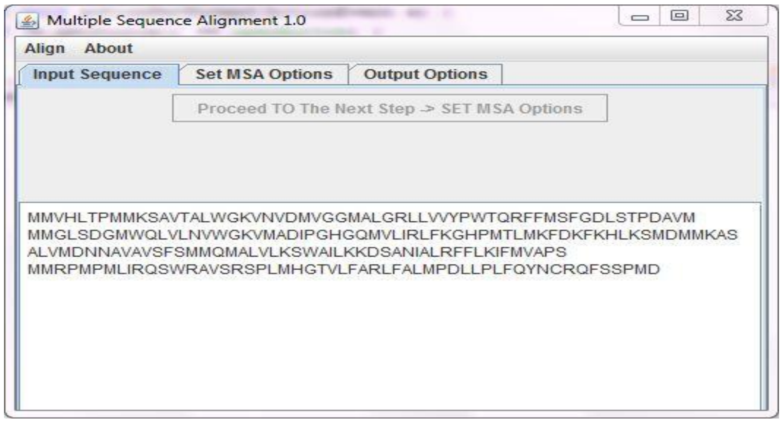
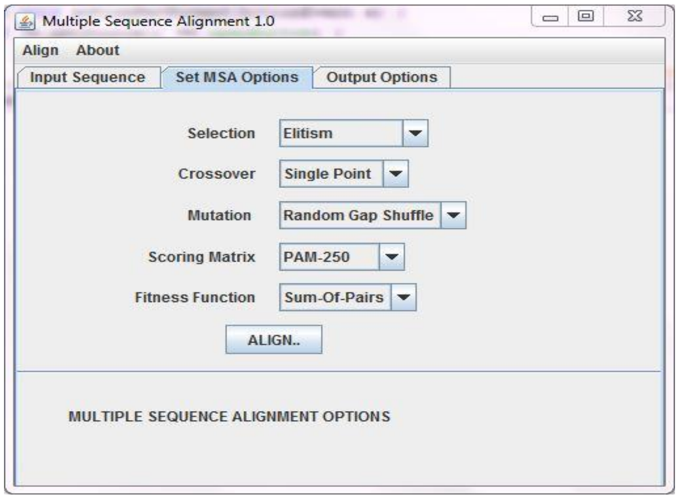


Figure 1: Interface for setting up the various GA operators


Multiple Sequence Alignment 1.0
Align
About
Input Sequence
Set MSA Options
Output Options
Proceed TO The Next Step -> SET MSA Options

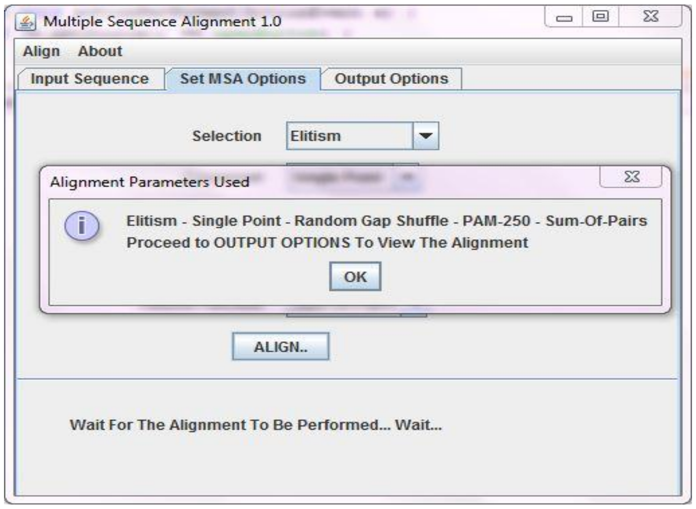
```
MMVHLTPMMKSAVTALWGKVNVDMVGGMALGRLLVVYPWTQRFFMSFGDLSTPDAVM
MMGLSDGMWQLVLNVWGKVMADIPGHGQMVLIRLFKGHPMTLMKFDKFKHLKSMDMMKAS
ALVMDNNAVAVSFSMMQMALVLKSWAILKKDSANIALRFFLKIFMVAPS
MMRPMPMLIRQSWRAVSRSPLMHGTVLFARLFALMPDLLPLFQYNCRQFSSPMD
```

Figure 2: Input sequences can be seen here


Multiple Sequence Alignment 1.0
Align
About
Input Sequence
Set MSA Options
Output Options
ALIGN..
MULTIPLE SEQUENCE ALIGNMENT OPTIONS

Selection: Elitism
Crossover: Single Point
Mutation: Random Gap Shuffle
Scoring Matrix: PAM-250
Fitness Function: Sum-Of-Pairs

Figure 3 (a)



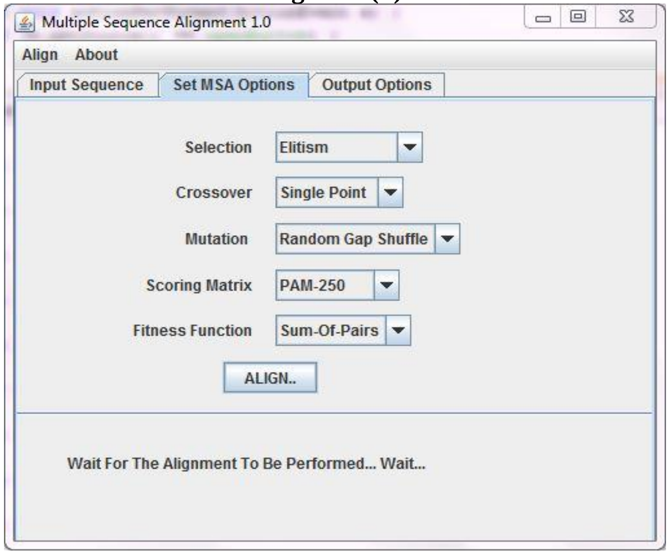
Selection: Elitism

Alignment Parameters Used

Elitism - Single Point - Random Gap Shuffle - PAM-250 - Sum-Of-Pairs
Proceed to OUTPUT OPTIONS To View The Alignment

Wait For The Alignment To Be Performed... Wait...

Figure 3 (b)

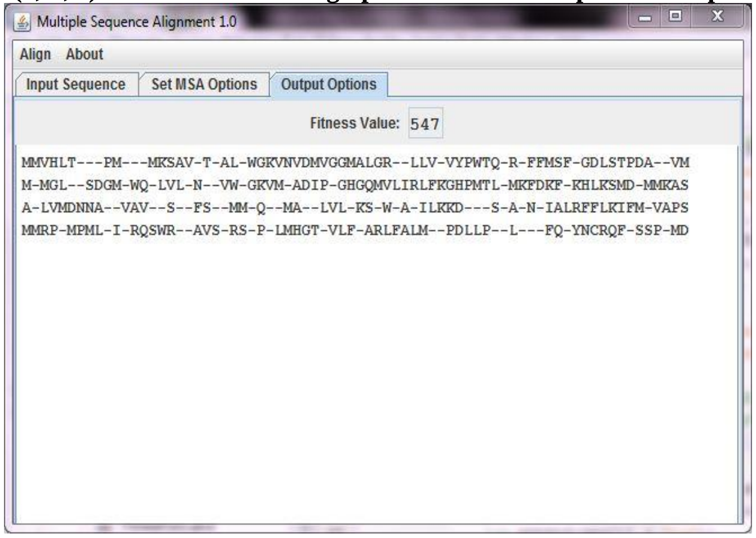


Selection: Elitism
Crossover: Single Point
Mutation: Random Gap Shuffle
Scoring Matrix: PAM-250
Fitness Function: Sum-Of-Pairs

Wait For The Alignment To Be Performed... Wait...

Figure 3 (c)

Figure 3(a, b, c): Interface for setting up the various GA operators & parameters



Fitness Value: 547

```
MMVHLT---PM---MKSAV-T-AL-WGKVNVDMVGGMALGR--LLV-VYPWTQ-R-FFMSF-GDLSTPDA--VM
M-MGL--SDGM-WQ-LVL-N--VW-GKVM-ADIP-GHGQMVLIRLFKGHPMTL-MKFDKF-KHLKSMD-MMKAS
A-LVMDNNA--VAV--S--FS--MM-Q--MA--LVL-KS-W-A-ILKKD---S-A-N-IALRFFLKIFM-VAPS
MMRP-MPML-I-RQSWR--AVS-RS-P-LMHGT-VLF-ARLFALM--PDLLP--L---FQ-YNCRQF-SSP-MD
```

Figure 4: Interface showing the aligned sequences.

IV. Results

No. of Generations	No. of Sequences				
	5	10	20	50	75
1	-2890	-38009	-149889	-770641	-1834804
4	-2853	-37781	-149203	-770055	-1833737
30	-2800	-37781	-149029	-769762	-1833668
50	-2735	-37781	-148643	-769762	-1833579
100	-2712	-37781	-148373	-769762	-1833149
500	-2590	-37544	-148228	-	-
1000	-2570	-37544	-148101	-	-
2000	-2558	-37348	-147948	-	-
5000	-2527	-37348	-147880	-	-
10000	-2515	-37348	-147778	-	-

Table 1: Showing the variations in Fitness Score on changing the Number of Sequences to be aligned. (Results taken, while using Single- Point Crossover, Gap Penalty=-3 as the remaining Input Parameters in the algorithm)

From the above obtained table (table 1) it can be observed that the fitness value tend to improve or converge in each successive generation.

Gen. No.	gap- Shuffle	random-insert-delete
1	-9553	-9526
5	-9469	-9526
30	-9446	-9474
50	-9363	-9458
100	-9363	-9362
500	-9340	-9362
1000	-9240	-9245

Table 2: Showing the variations in Fitness Score on changing the Mutation method (Results taken, while using Six input sequences, Single- Point Crossover, Gap Penalty=-2 as the remaining Input Parameters in the algorithm)

Gen. No.	Single-point	Double-point	min-max
1	-9567	-9471	-9571
5	-9523	-9324	-9346
30	-9473	-9291	-9346
50	-9473	-9204	-9346
100	-9416	-9204	-9346
500	-9291	-9182	-9346
1000	-9291	-9176	-9286

Table 3: Showing the variations in Fitness Score on changing method of Crossover. (Results taken, while using Six input sequences, Gap Shuffle Mutation, PAM-250 matrix as the remaining Input Parameters)

Gen. No.	Gap Penalty of -4	Gap Penalty of -3	Gap Penalty of -2	Gap Penalty of -1
1	-1375	-1149	-363	129
5	-1375	-1087	-363	129
30	-1311	-749	109	693
50	-1311	-712	109	810
100	-1060	-229	331	810
500	-932	-94	331	937
1000	-932	-94	331	938
2000	-932	2	429	1090
5000	-844	2	464	1203
10000	-712	2	464	1203
20000	-586	64	607	1203
30000	-559	297	617	1278

Table 4: Showing the variations in Fitness Score on changing gap penalties:
(Results taken, while using Four input sequences, Single- Point Crossover, Gap Shuffle Mutation, BLOSUM-45 matrix as the remaining Input Parameters)

Gen. No.	Gap Penalty of -4	Gap Penalty of -3	Gap Penalty of -2	Gap Penalty of -1
1	-1884	-1215	-580	-151
5	-1884	-1073	-471	254
30	-1525	-872	-416	254
50	-1525	-872	-230	650
100	-1408	-849	-230	650
500	-1364	-776	-130	650
1000	-1364	-776	44	650
2000	-1146	-652	44	650
5000	-1010	-563	139	777
10000	-1010	-388	380	811
20000	-1010	-388	380	1165
30000	-1010	-388	380	1165

Table 5: Showing the variation in Fitness Score on changing gap penalties:
(Results taken, while using Four input sequences, Single- Point Crossover, Gap Shuffle Mutation, PAM-250 matrix as the remaining Input Parameters)

V. Conclusion

We've used various methods of crossover, mutation and selection schemes for multiple alignment. The results of each alignment tend to improve, which is being shown by the increasing fitness value with increase in number of iterations.

VI. References

1. Lecture notes on 'Sequence Alignment' by Kun-Mao Chao, Department of Computer Science and Information Engineering National Taiwan University (2005)
2. Goldberg, D.E. (1989). *Genetic algorithms in search, optimization & machine learning.* Reading, MA: Addison-Wesley Publishing Company, Inc.
3. Hernandez, D, Grass, R., and Appel, R, (2004). MoDEL: an efficient strategy for ungapped local multiple alignment. *Computational Biology and Chemistry, 28, 119-128.*
4. Horng, J.T., Wu, L.C., Lin CM, and Yang, B.H. (2005). A genetic algorithm for multiple sequence alignment. *Soft Computing, 9, 407-420.*
5. Wang, C, and Lefkowitz, E.J. (2005). Genomic multiple sequence alignments: Refinement using a genetic algorithm. *BMC Bioinformatics, 6:200.*
6. Shyu, C, Sheneman, L., and Foster, J.A. (2004). Multiple sequence alignment with evolutionary computation. *Genetic Programming and Evolvable Machines,5, 121-14.*
7. Buscema, M. (2004). Genetic doping algorithm (GenD): Theory and applications. *Expert Systems, 21(2),* 63-79.
8. Notredame C, & Higgins, D.G. (1996). SAGA: Sequence alignment by genetic algorithm. *Nucleic Acids Research, 24, 8, 1515-1524.*
9. Segun A Fatumo, Ibidapo O Akinyemi, and Ezekiel F Adebiyi: Aligning Multiple Sequences with Genetic Algorithm, International Journal of Computer Theory and Engineering, Singapore, Vol. 1 No. 2, [186-190], 2009.
10. H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology", Siam J. Appl. Math., vol. 48, no. 5, pp. 1073-1082, October 1988.
11. K. Kosmas and H. K. Donald. Genetic Algorithms and the Multiple Sequence Alignment Problem in Biology (1996). Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, February, Baton Ronge, LA.
12. S. F. J. Altschul. "Gap Costs for Multiple Seqence Alignment" (1989) Theoretical Biol., Vol 138, pp 297 – 309.
13. Amouda Nizam, Buvaneswari Shanmugham, Kuppuswami Subburaya, Self-Organizing Genetic Algorithm for Multiple Sequence Alignment, GJCST (2011), Volume 11, Issue 7, 7-14
14. Yang Chen, Jinglu Hu, Member, IEEE, Kotaro Hirasawa, Member, IEEE, Songnian Yu. (2008). Multiple Sequence Alignment Based on Genetic Algorithms with Reserve Selection ICNSC, pp 1511-1516.