Multiple Sequence Alignment in Bioinformatics.

Harshita Badlani, Abhinav Sinha, Abhinav Mittal, Alok Kumar (Asst. Prof., GCET)
Galgotias college of Engineering and Technology, Greater Noida.

Abstract

This paper proposes few genetic operators to obtain better alignments of multiple molecular sequences. 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. These fitness values then help to find heart and diabetes problems for that chromosome. 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.

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. Proposed algorithm 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. On the basis of this fitness value, probability of heart and diabetes problems is also found for the next 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.
3. Probability of heart and diabetes problems in the next generation based on their fitness value.

The objective of this paper is to find the rules for finding fitness of any person based on his/her parents DNA and examining the probability of Heart or Diabetes diseases.

Related Work:-

Genetic algorithm (GA) refers to a model introduced and investigated by John Holland in 1975 for adaptation processes of nature. Generally stated, a GA is any population based model that uses selection and recombination operators to generate new sample points in a search space. GA computationally utilizes a natural evolutionary process similar to the process first described by Charles Darwin in his "The Origin of Species", to solve a given problem. GA is a global search procedure that searches from one population of points to another. GA is a
probabilistic search procedure, which is being frequently applied to difficult optimization and learning problems. There are two versions of the GA, namely the natural GA and the computational GA.

Genetic algorithms were inspired by the processes observed in natural evolution. They attempt to mimic these processes and utilize them for solving a widerange of optimization problems. In general, genetic algorithms perform directed random searches through a given set of alternatives with respect to the given criteria of goodness. These criteria are required to be expressed in terms of an objective function, which is usually referred to as a fitness function.

Genetic algorithms require that the set of alternatives to be searched through be finite. If we want to apply them to an optimization problem where this requirement is not satisfied, the set involved and select an appropriate finite subset. It is further required that the alternatives be coded in strings of some specific finite length which consist of symbols from some finite alphabet. These strings are called chromosomes, the symbols that form them are called genes, and their set is called a gene pool. Genetic algorithms search for the best alternative in the sense of a given fitness function through chromosomes evolution.

Genetic Algorithms search for the best alternative (in the sense of a given fitness function) through chromosomes’ evolution. Basic steps in genetic algorithms figure. First, an initial population of chromosomes is randomly selected. Then each of the chromosomes in the population is evaluated in terms of its fitness (expressed by the fitness function).

Next, a new population of chromosomes is selected from the given population by giving a greater change to select chromosomes with the high fitness. This is called natural selection. The new population may contain duplicates. If given stopping criteria (e.g., no change in the old and new population, specified computing time, etc.) are not met, some specific, genetic – like operations are performed on chromosomes of the new population. These operations produce new chromosomes, called offspring’s. The same steps of this process, evaluation and natural selection, are then applied to chromosomes of the resulting population. The whole process is repeated until given stopping criteria are met. The solution is expressed by the best chromosome in the final population.

There are many variations on these basic ideas of genetic algorithms. To describe a particular type a genetic algorithm is greater detail, let G denote the gene pool, and let n denote the length of strings of genes that form chromosome. That is, chromosomes are n - tuples in Gn. The size of the population of chromosomes is usually kept constant during the execution of genetic algorithm. That is, when new members are added to the population, the corresponding number of old members are excluded. Let m denote this constant population size.

Proposed Work:

The pseudo code is as follows:

1. Start
2. Initialization: Sequence 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.
5. 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 chromosomes using crossover and mutation schemes. Here we have experimented with a simple scheme where the chromosomes produced through the initial fitness value less than the parent chromosomes, are discarded i.e., the best two chromosomes of parent i, parent j, child i and child j. One & two-point crossover schemes are tried.

Mutation operations 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 and its associated parameters.

Result:- The best sequence alignment would correspond to the chromosome with highest fitness value after N generations are done. The desired minimum acceptable score is obtained.

End

Example Demonstration: INPUT SEQUENCE
>MMVHLPMMKSAVTLWGVKNVDMVGAMMLGRLLNYVPWTQRFMSFGDLSTPDAVM
>MMGLSDGMWQLVNWGKVMAIPGHGQMVLIRLFKGPMTLMKFDKFKHLKSMMDMKAS
>ALVMDDNNAVSVSFMQMOLALVKSAILKDSANIALRFFLKFIMVAPS
>MMRPMPMLIRQWSVRSPLMHGTVLFARLFALMPDLLPLFQYNYCRQFSSPM

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, leni: length of input sequence, iPop: initial population set)
{
    Calculate the length of sequence using length(N) = 1.2 * lmax
    Fork = 1 to iPop
    
    For each sequence i = 1 to N
    compute the gap to be inserted as gap(i) = length - leni.
    
    Foreach sequence i = 1 to N
    insert gap
    number of gaps at random positions.
    this_initial_alignment is referred using seq-alk.
}
```

lmax = 61, corresponding to the longest sequence
length(N) = 1.2 * lmax, gap1 = length - len1 = 17, gap2 = length - len2 = 13, gap3 = length - len3 = 25, gap4 = length - len4 = 20

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

```
>MMVHLP --PM--MKSAV --TALWGVKNVDMVGAMMLGRLLNYVPWTQRFMSFGDLSTPDA--VM
M-MGLSDGMWQLVNWGKVMAIPGHGQMVLIRLFKGPMTLMKFDKFKHLKSMMDMKAS
>ALVMDDNNAVSVSFMQMOLALVKSAILKDSANIALRFFLKFIMVAPS
>MMRPMPMLIRQWSVRSPLMHGTVLFARLFALMPDLLPLFQYNYCRQFSSPM
```

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.

Crossover
Crossover is a process of taking more than one parent chromosome 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-mincrossover. Crossover 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 random number determined based on the length of the parents. Each such crossover results into two child chromosomes.
mes. As an experimental scheme, we have restored only those child chromosomes which have better fitness scores than their parents.

**Mutation**
Mutation is a genetic operator used to maintain genetic diversity from one generation 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 the initial population set are selected and mutation is applied upon them.

**GapShuffling**
Gap-Shader-Mut(chrom-popm: chromosome generation of m chromosomes, mutP: mutation probability)

```{fork=1 to m representing each chromosome, perform operations on decoded alignment sequence, seq_alk}
{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. }
```

**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 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. During fitness evaluation, all fully-gapped columns in an alignment are ignored.

**Acknowledgment:**
It is a great opportunity for us to write this paper. At the time of preparing this paper, we have gone through different books and websites.

We acknowledge with gratitude to our assistant professor Alok Kumar, who has always been sincere and helpful in making us understanding different problems.

**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.

**References:**
8. Lecture notes on Sequence Alignment by Kun-Mao Chao, Department of Computer Science and Information Engineering, National Taiwan University (2005).