Solving Multiple Sequence Alignment Problems using Various Evolutionary Algorithm

Farah Nazifa ID: 10201032

A DISSERTATION

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

Supervisor: Farzana Rashid

Co-supervisor: Dilruba Showkat

May 2015

Table of Contents Page

STATEMENT

We hereby proclaim that this thesis is based on the results we found by means of our effort. Contents of the work found by other researcher(s) are motioned by references. Previously researchers worked on this piece of work but we are verifying their results in comparison to the standard tool used of our project content.

Signature of Author:

____

Farah Nazifa

10201032

Signature of Thesis Supervisor:

____

Farzana Rashid

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to Almighty Allah (SWT) who gave me the opportunity, determination, strength and intelligence to complete my thesis.

I want to acknowledge my junior, Saddat Hasan, who has consistently shown his interest in my work and assisted me in its completion.

I am blessed to be always supported by my parents, Md. Enayet Karim and Fatema Akter, my best friend, Fabia and my aunt, Mila for their overwhelming love and care.

I would like to thank my supervisor Farzana Rashid and co-supervisor Dilruba Showkat sincerely for their consistent supervision, guidance and unflinching encouragement in accomplishing my work.

Abstract

Bioinformatics is an art and science conceptualized with the use of computational biology. This review paints a broader picture of bioinformatics, on how the sequences of DNA, RNA are organized in the right possible alignment with minimal gap between the sequences. The roles of bioinformatics are highlighted at multiple points along the path from high-tech data generation to biological discovery. Sequence Alignment is used to find the areas of sequence similarity that could point to the structure of an evolutionary ancestor or provide information about the evolutionary history of the sequences. Multiple Sequence Alignment is a way of arranging the sequences of DNA and RNA to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. The work will involve study of multiple algorithms, Genetic Algorithm, Artificial Bee Colony Algorithm and Hybrid Algorithm. The proposed algorithms are implemented for solving the problem, Multiple Sequence Alignment. Result based on fitness against number of iterations graphical representation will show the best algorithm for this particular problem. However, results, similarities, differences efficiency of the implementations on these algorithms will be compared.

Part 1: Introduction

1.1 Objective:

Bioinformatics is hypothesizing biology in terms of molecules (in the sense of physical chemistry) and then applying informatics techniques (derived from disciplines such as applied math, CS, and statistics) to understand and organize the information associated with these molecules, on a large-scale.

DNA, or deoxyribonucleic acid, is the hereditary material in humans and almost all other organisms. Nearly every cell in a person's body has the same DNA. RNA or ribonucleic acid is a ubiquitous family of large biological molecules that perform multiple vital roles in the coding, decoding, regulation, and expression of genes. DNA and RNA are two different nucleic acids found in the cells of every living organism. Both have significant roles to play in cell biology. DNA and RNA structure are similar because they both consist of long chains of nucleotide units. DNA contains the genetic information of an organism, and this information dictates how the body's cells would construct new proteins according to the genetic code of the organism.

To know the unknown properties of certain DNA, scientist has to match it to a known DNA to observe similarities and dissimilarities. This is known as DNA sequencing. However, matching DNA sequences is not a job for human being as there is lot to calculate and remember. Here, bio-informatics comes to play. Many algorithms have been proposed to solve DNA sequencing efficiently. In this paper, I am going to propose a new way to solve DNA sequencing. I will describe and explain my findings on using Artificial Bee Colony Algorithm on multiple DNA sequencing problem.

Part 2: Theory

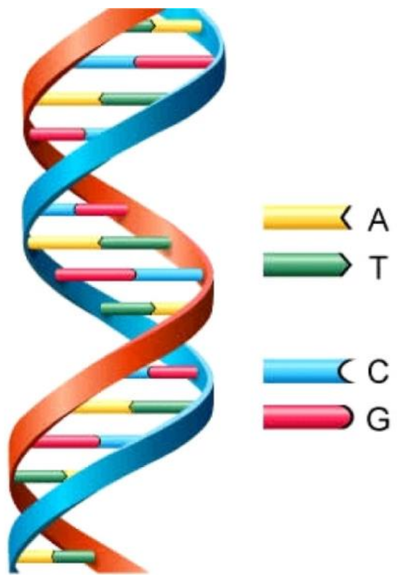
2.1 Deoxyribonucleic Acid (DNA):

We all observe that there are certain common physical and behavioral characteristics among the members of same family. The reason lies in a molecule called **deoxyribonucleic acid**, which contains the biological instructions that make each species unique. DNA, along with the instructions it contains, is passed from adult organisms to their offspring during reproduction.

Researchers refer to DNA found in the cell's nucleus as nuclear DNA. An organism's complete set of nuclear DNA is called its genome. In sexual reproduction, organisms inherit half of their nuclear DNA from the male parent and half from the female parent. However, organisms inherit their entire mitochondrial DNA from the female parent. This occurs because only egg cells, and not sperm cells, keep their mitochondria during fertilization.

DNA is made of chemical building blocks called nucleotides. These building blocks are made of three parts: a phosphate group, a sugar group and one of **four types of nitrogen bases**. To form a strand of DNA, nucleotides are linked into chains, with the phosphate and sugar groups alternating.

The four types of nitrogen bases found in nucleotides are **adenine** (A), **thymine** (T), **guanine** (G) and **cytosine** (C). The order, or sequence, of these bases determines what biological instructions are contained in a strand of DNA. For example, the sequence ATCGTT might instruct for blue eyes, while ATCGCT might instruct for brown.



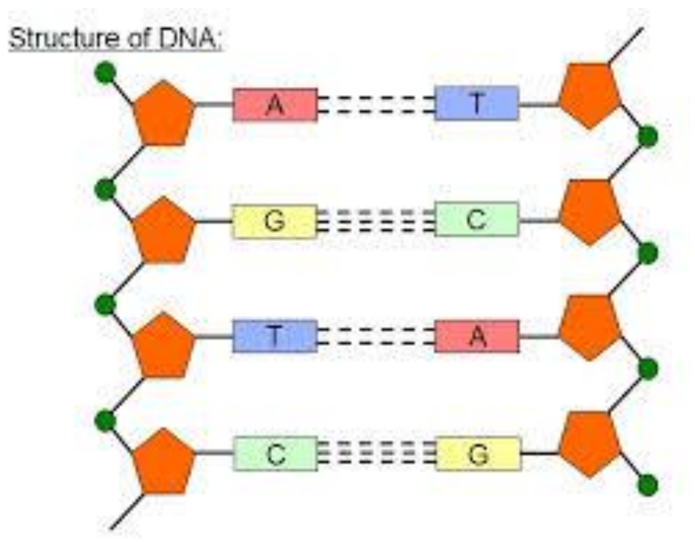


Figure 1: Structure of DNA with nitrogen bases

Figure 2: Bases position in DNA

Scientist use the term "double helix" to describe DNA's winding, two-stranded chemical structure. This shape - which looks much like a twisted ladder - gives DNA the power to pass along biological instructions with great precision.

Because of the highly specific nature of this type of chemical pairing, base A always pairs with base T, and likewise C with G. So, if you know the sequence of the bases on one strand of a DNA double helix, it is a simple matter to figure out the sequence of bases on the other strand.

DNA's unique structure enables the molecule to copy itself during cell division. When a cell prepares to divide, the DNA helix splits down the middle and becomes two single strands. These single strands serve as templates for building two new, double-stranded DNA molecules - each a replica of the original DNA molecule. In this process, an A base is added wherever there is a T, a C where there is a G, and so on until all of the bases once again have partners.

In addition, when proteins are being made, the double helix unwinds to allow a single strand of DNA to serve as a template. This template strand is then transcribed into mRNA, which is a molecule that conveys vital instructions to the cell's protein-making machinery.

2.2 Ribonucleic Acid (RNA):

RNA molecules are single stranded nucleic acids composed of nucleotides. RNA plays a major role in protein synthesis as it is involved in the transcription, decoding, and translation of the genetic code to produce proteins. Like DNA, RNA nucleotides contain three components: a Nitrogenous Base, a Five-Carbon sugar, a Phosphate Group. RNA nitrogenous bases include adenine (A), guanine (G), cytosine (C) and uracil (U). The five-carbon (pentose) sugar in RNA is ribose.

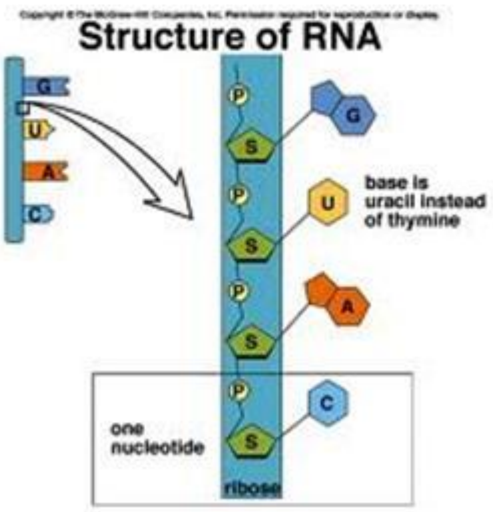


Figure 3: Structure of RNA

2.3 Sequence Alignment (SA)

When DNA or RNA sequences are placed together, it is seen that there reflects an interesting effect of alignment which is known as sequence alignment. Many bioinformatics tasks depend upon successful alignments.

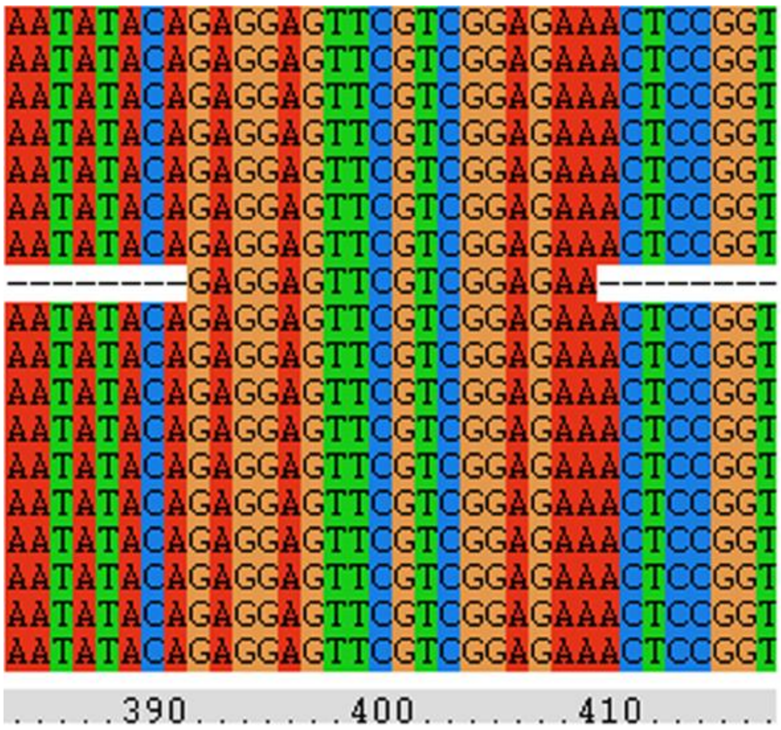
In a symbolic sequence, a letter signifies each base or residue monomer in each sequence. The conventional idea is to print the lone-letter codes for the constituent monomers in order to a fixed font (from the N-most to C-most end of the DNA sequence). However, this is based on the assumption that the combined monomers evenly spaced along the single dimension of the molecule's primary structure. From now on, we will refer to an alignment of two DNA sequences.

Every element in a trace is either a match or a gap. Where a residue in one of two aligned sequences is identical to its counterpart in the other the corresponding amino-acid letter codes in the two sequences are vertically aligned in the trace: a match. When a residue in one sequence seems to have been deleted since the assumed divergence of the sequence from its counterpart, its "absence" is labelled by a dash in the derived sequence. When a residue appears to have been inserted to produce a longer sequence a dash appears opposite in the unaugment sequence. Since these dashes represent "gaps" in one or other sequence, the action of inserting such spacers is known as gapping.

A deletion in one sequence is symmetric with an insertion in the other. When one sequence is gapped relative to another a deletion in sequence a can be seen as an insertion in sequence b. Indeed, the two types of mutation are referred to together as indels. If we imagine that at some point one of the sequences was identical to its primitive homologue, then a trace can represent the three ways divergence could occur (at that point).

2.4 Multiple Sequencing Alignment (MSA)

A multiple sequence alignment (MSA) is a sequence alignment of three or more biological sequences, generally protein, DNA, or RNA. In many cases, [1] it plays a crucial protagonist in distinguishing regions of significant sequence match from collections of primary sequences of nucleic acid or proteins. MSA can also be used to funding the reconstruction of phylogenetic trees, find designs to illustrate protein families, detect homology between new sequences and existing ones, and foresee the secondary and tertiary structure of protein sequences. The foundation of sequence alignment and MSA is evolutionary concept. According to this theory, during the course of evolution, mutations occur, and differences are created among families of species. Most of these changes are due to local mutations which consist of three operations, including insertion (inserting certain letters to the sequences), deletion (deleting certain letters from the sequences), and substitution (replacing a letter by another). [2]



```
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
--------GAGGAGTTCGTCGGAGAA--------
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
AATATACAGAGGAGTTCGTCGGAGAAACTCCGGT
.....390.......400.......410......
```

Figure 4: MSA of DNA sequences

Given two sequences X and Y, a pairwise alignment indicates positions of each sequence that are considered to be functionally or evolutionarily related. Given a family of N sequences, we would like to find out the common patterns of this family. Since aligning each pair of sequences from S separately often does not reveal the common information, it is necessary to perform multiple sequence alignment (MSA). Within an MSA, preserved motifs often appear as

S S S = (,...,) 1 N columns with a much lower degree of variation than their surroundings.

2.5 Why do we need MSA?

Typically, with a pairwise alignment, we infer biological relationships from the sequence similarity. With a multiple alignment, we know that the sequences are biologically related, and we use the multiple alignment to find the areas of sequence similarity that could point to the structure of an evolutionary ancestor or provide information about the evolutionary history of the sequences. Multiple sequence alignments, or MSAs, are also more sensitive to sequence similarities than a pairwise alignment because the conserved regions could be so dispersed that a pairwise alignment would not find them.

MSAs allow us to:

- infer phylogenetic relationships
- elucidate biological facts about proteins since most conserved regions are biologically significant
- formulate and test hypotheses about protein 3-D structure and function

2.6 Scoring Function

The first scoring function is the minimum entropy scoring function. The goal of this scoring algorithm is to minimize the entropy, or randomness in the alignment. To calculate the entropy of the alignment, first, we must calculate the probability of column i and then use that probability to calculate a score for that column. This score measures the variability observed in the aligned column i. By minimizing the sum of this column score over all of the columns, we minimize the entropy and create a “good” alignment.

Another scoring algorithm is the sum of pairs algorithm. In this scoring algorithm, the score of an MSA is the sum of the scores of all of the pairwise alignments.

The sum-of-pairs (SP) score of a multiple alignment m is the sum of the scores of all induced pairwise alignments.

SP score for column mi is:

$$S\ (m_i) = \sum_{k<l} s(m_i^k, m_i^l),\ s(a,b) \text{ is obtained from substitution matrix}$$

Taking look at the methods that use these functions:

- Dynamic Programming
- Heuristic
- Progressive
- Progressive with refinement
- Model or profile alignment
- Genetic Algorithm
- Particle Swarm Optimization
- Artificial Bee Colony

In summary, choosing the best scoring scheme is critical to the creation of a meaningful MSA. Dynamic programming methods, while guaranteeing an optimal alignment, are too computationally expensive to use for even moderate numbers of sequences, although there are heuristics that can be used to reduce the number of calculations. Progressive methods are much less computationally expensive, but they are very sensitive to initial alignments and may not produce good alignments, especially for distantly related sequences. Using an iterative approach improves the alignments produced by progressive methods, but it is still sensitive to the initial alignment. The profile methods allow integration of position-specific information and profile-profile alignments. In general, most computational methods use a large number of heuristics to obtain an optimal alignment.

Part 3: Proposed Evolutionary Algorithm

3.1 Genetic Algorithm:

Genetic Algorithms was invented to imitate some of the courses observed in natural evolution. Many people, biologists included, are astonished that life at the level of complexity that we observe could have evolved in the relatively short time suggested by the fossil record. The idea with GA is to use this power of evolution to solve optimization problems. The father of the original Genetic Algorithm was John Holland who invented it in the early 1970's. [3]

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such, they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomized, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution; especially those follow the principles first laid down by Charles Darwin of "survival of the fittest.” In nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones. [4]

3.1.1 Overview of GA

GA simulates the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA. Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution. GAs are based on an analogy with the genetic structure and behavior of chromosomes within a population of individuals using the following foundations:

i. Individuals in a population compete for resources and mates.
ii. Those individuals most successful in each 'competition' will produce more offspring than those individuals that perform poorly.
iii. Genes from `good' individuals propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent.
iv. Thus each successive generation will become more suited to their environment.

There are five main aspects of GA. They are namely

i. Population
ii. Fitness calculation
iii. Selection
iv. Crossover
v. Mutation

3.1.2 Population

A population of individuals is maintained within search space for a GA, each representing a possible solution to a given problem. Each individual is coded as a finite length vector of components, or variables, in terms of some alphabet, usually the binary alphabet {0, 1}. To continue the genetic analogy these individuals are likened to chromosomes and the variables are analogous to genes. Thus, a chromosome (solution) is composed of several genes (variables). A fitness score is assigned to each solution representing the abilities of an individual to `compete'. The individual with the optimal (or generally near optimal) fitness score is sought. The GA aims to use selective `breeding' of the solutions to produce `offspring' better than the parents do by combining information from the chromosomes. In the figure below the parent1 and parent, 2 are the population.

Fitness Calculation: Fitness calculation is a way to determine which candidates of the population serves the purpose best. For instance, in case of TSP, the way to calculate fitness is to determine which route in the population cost least. For different problems, the method to calculate fitness value changes accordingly.

3.1.3 Selection

There can be more the two individuals in population. Then certain number of candidate must be chosen for the crossover. This is called the selection. The selection process is based on the fitness calculation. It gives preference to better individuals, allowing them to pass on their genes to the next generation. The goodness of each individual depends on its fitness.

Fitness may be determined by an objective function or by a subjective judgment.

Crossover: Crossover refers to merging of two or more parent individuals to make new offspring. In crossover, a portion of uniform size is selected from both the parents. Then the portion from first parent is added to the portion of second, thus an offspring is born. Similarly second child is produced. Prime distinguished factor of GA from other optimization techniques. Two individuals are chosen from the population using the selection operator. A crossover site along the bit strings is randomly chosen. The values of the two strings are exchanged up to this point

If S1=000000 and s2=111111 and the crossover point is 2 then S1'=110000 and s2'=001111, the two new offspring created from this mating are put into the next generation of the population. By recombining portions of good individuals, this process is likely to create even better individuals

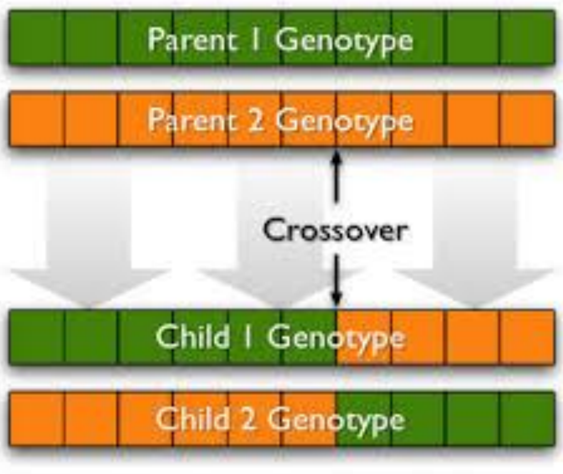


Figure 5: GA crossover

3.1.4 Mutation

After Crossover has run several times and yet expected result is not found, then mutation comes into action. Mutation refers to changing each candidate parent at certain point or points. With some low probability, a portion of the new individuals will have some of their bits flipped. Its purpose is to maintain diversity within the population and inhibit premature convergence. Mutation alone induces a random walk through the search space. Mutation and selection (without crossover) creates parallel, noise-tolerant, hill-climbing algorithms.

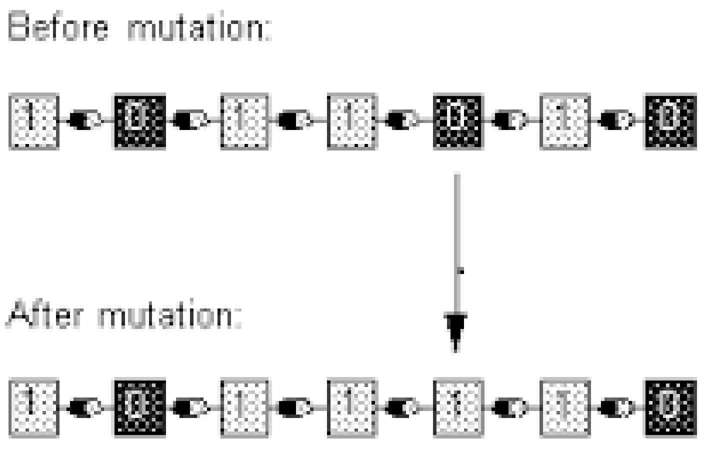


Figure 6: GA mutation

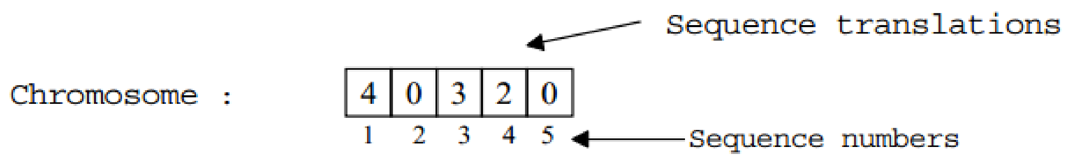
3.1.5 Solving MSA with GA

Genetic Algorithm solves MSA by two methods. One method solves MSA without gaps; the other solves MSA considering the gaps.

- **MSA without gaps**

Each likely alignment can be denoted as a chromosome, which is composed of N genes, where N is the number of sequences to be aligned. Every gene saves the translation of its corresponding order from left to right.

Example:



```
Corresponding alignment:     - - - - K L G Q G C F
                             G E K F K Q G
                             - - - G F E G N I I
                             - - D T Y E N T P
                             L L G S G S I
```

The fitness function can be as simple as calculating the total number of matching symbols or as complex as considering the type of symbols bring into line, their place in the sequences, their adjoining symbols, etc. In this application, the simplest possible fitness function is used, which is to sum the total number of matches and then assign 1 point for each match:

$$\text{Fitness} = (\text{total matches}) * 1.0$$

• MSA with Gap

This time every alignment is denoted by a chromosome composed of G*N genes where G is the maximum number of gaps allowed in a sequence and N is the number of input sequences to be aligned. Every gene saves the position in which a gap appears in its corresponding sequence. The location of the gene in the chromosome denotes the sequence with which it is associated. [5]

For MSA with gaps, the fitness function is modified to include some penalty points for the gaps:

$$\text{Fitness} = (\text{total matches}) * 1.0 - (\text{gap penalties})$$

Each matching pair of symbols adds 1 point to the fitness value, where 4 points are subtracted for every group of successive gaps. Also 0.4 points are deducted for each individual gap. This distinction between "gap groups" and "individual gaps" helps to impose a start-up penalty for introducing new gaps.

$\text{gap penalties} = (\text{gap groups}) * 4.0 + (\text{total number of gaps}) * 0.4$

3.2 Artificial Bee colony Algorithm

Artificial Bee Colony (ABC) is a quite a new member of swarm intelligence. ABC attempts to model natural behavior of honeybees in food searching. Honeybees use quite a few mechanisms like “waggle dance” to optimally locate food sources and to search new ones. This makes them a good contender for developing new intelligent search algorithms. By performing waggle dance, successful foragers share the information about the direction and distance to areas of flower and the amount of nectar within this flower with their hive mates. So, this is an effective mechanism in which foragers can recruit other bees in their colony to productive locations to collect resources. Bee colony can rapidly adjust its searching pattern in time and space according to changing nectar sources with an accurate outcome.

Studies on dancing behavior of bees show that while performing the waggle dance, the direction of bees indicates the direction of the food source in relation to the Sun, the intensity of the waggles indicates how far away it is and the duration of the dance indicates the amount of nectar on related food source. Waggle dancing bees that have been in the hive for an extended time adjusts the angles of their dances to accommodate the changing direction of the sun. Therefore, bees that follow the waggle run of the dance are still correctly led to the food source even though its angle relative to the sun has changed. So collective intelligence of bees based on the synergistic information exchange during waggle dance.

When a bee has found a promising food resource, it will return to the hive and communicate its location by shuffling in a direction, which relates to the sun's position. So, for example, if the supplies can be found by flying in the direction of the sun, the bee will record its whereabouts by waggling its body straight up. Yet, if it is located directly to the left of the hive, the creature will dance at a 90-degree angle to the sun.

3.2.1 How ABC works

Social insect colonies can be well thought of as dynamical system collecting information from environment and changing its actions accordingly. While collecting information and adjustment courses, individual insects do not perform all the errands because of their specializations. Usually, all social insect colonies behave according to their own division of labors related to their morphology. [4] Bee system consists of two essential components:

- Food Sources: The importance of a food source depends on different parameters such as its closeness to the nest, richness of energy and ease of extracting this energy. [5]
- Foragers:
    - Unemployed foragers: If it is assumed that a bee have no knowledge about the food sources in the search field, bee initializes its search as an unemployed forager. There are two possibilities for an unemployed forager-
    - Scout Bee: If the bee starts searching spontaneously without any knowledge, it will be a scout bee. The percentage of scout bees varies

from 5% to 30% according to the information into the nest. The mean number of scouts averaged over conditions is about 10%.

- Recruit: If the unemployed forager attends to a waggle dance done by some other bee, the bee will start searching by using the knowledge from waggle dance.

- Employed foragers (EF in Figure): When the recruit bee finds and exploits the food source, it will raise to be an employed forager who memorizes the location of the food source. After the employed foraging bee loads a portion of nectar from the food source, it returns to the hive and unloads the nectar to the food area in the hive. There are three possible options related to residual amount of nectar for the foraging bee. If the nectar amount decreased to a low level or exhausted, foraging bee abandons the food source and become an unemployed bee. If there are still sufficient amount of nectar in the food source, it can continue to forage without sharing the food source information with the nest mates or it can go to the dance area to perform waggle dance for informing the nest mates about the same food source. The probability values for these options highly related to the quality of the food source.

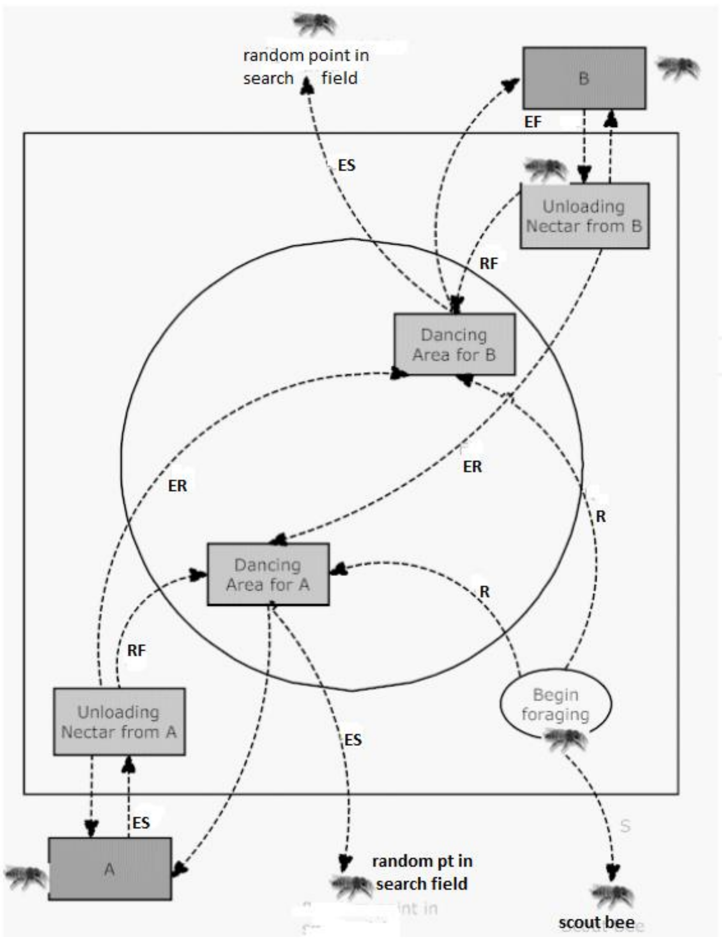
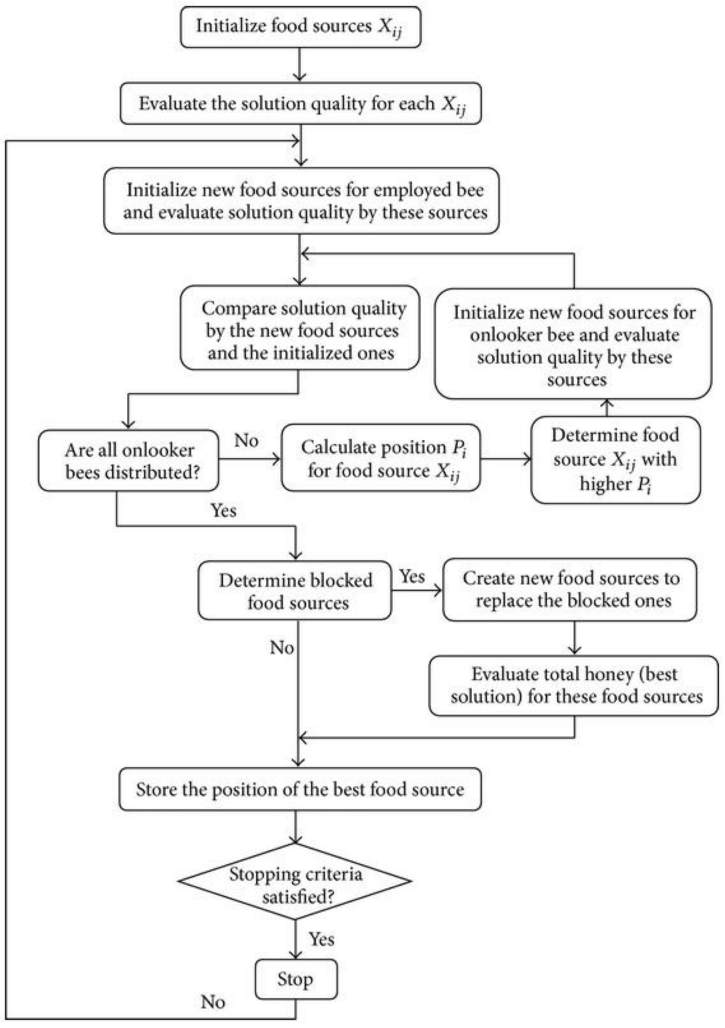


Figure 6: ABC

- Experienced foragers: These types of forager use their historical memories for the location and quality of food sources. It can be an inspector, which controls the recent status of food source already discovered. It can be a reactivated forager by using the information from waggle dance. It tries to explore the same food source discovered by itself if there are some other bees confirm the quality of same food source (RF in Figure S). It can be scout bee to search new patches if the whole food source is exhausted (ES in Figure). It can be a recruit bee, which is searching a new food source declared in dancing area by another employed bee (ER in Figure).

3.2.2 Flowchart of ABC


Initialize food sources X_{ij}
Evaluate the solution quality for each X_{ij}
Initialize new food sources for employed bee and evaluate solution quality by these sources
Compare solution quality by the new food sources and the initialized ones
Initialize new food sources for onlooker bee and evaluate solution quality by these sources
Are all onlooker bees distributed?
No
Calculate position P_i for food source X_{ij}
Determine food source X_{ij} with higher P_i
Yes
Determine blocked food sources
Yes
Create new food sources to replace the blocked ones
No
Evaluate total honey (best solution) for these food sources
Store the position of the best food source
Stopping criteria satisfied?
Yes
Stop
No

3.2.3 Solving MSA with ABC

The ABC algorithm is based on populations. The location of a food source embodies a likely solution of the optimization problem and the quantity of nectar represents the excellence of the proposed solution. The number of scouts is equal to the number of solutions in the population. The first step is to create arbitrarily a partitioned initial population. After the initialization, the population repeats the series of seeking employed bees, onlookers and scouts. The scout adjusts the locations of the sources in its memory and remembers the new position. [5]

In case of the new source if the nectar amount is greater than the previous, the scout remembers the position of the new source and forgets the old one. Otherwise, the scout preserves the location of current source in the memory. When all the scouts complete the search process, they share information about the locations of food sources with onlookers through waggle dance. Each onlooker calculates the information for the nectar in accordance to the dance of scouts and then chooses a food source according to the quantity of food in the source. The onlooker compares quantities of nectar in the new source with that in already stored. If the amount of the nectar is greater in the new source, the bee remembers the new position and forgets the old one. The scouts determine which sources should be abandoned and randomly select new sources. The employed bees search around for decisions.

The food source is presented by possible sequence alignments. The quality of solutions (nectar amount) is the evaluation function of the similarity between the sequences for a particular alignment. [7]

The proposed system is based on the idea of swarms and hives. The allocation of computing supercomputer resources is as follows: the entire super system simulates the behavior of a colony of beehives and the number of hives is equal to the number of computing nodes. Each computing node simulates the behavior of a hive. In a hive, there are q swarms, where q is the segments number of the super system. The swarms within a hive work on common lists of best temporary solutions and elite solutions.

Each hive has a queen bee, which gets the elite decision of all swarms on the hive. Scout 0 reads the sequences from the memory and stores it in sharing memory of the computational node (hive). Scouts generate initial solutions via sequence alignment including gaps. [6] Scouts then evaluate the quality of the alignment using the following method.

First, a calculation by columns is done – in the case of nucleotide sequences; the numbers of symbols – A, G, C and T – are counted. The numbers of symbols are associated and the nucleotides, which occur mostly in the different columns, are chosen. This is called the "nucleotide-favorite" sequence (fij). After the calculation of assessments in columns the so-called sequence-favorite contained in each location the corresponding favorite in the column is formed, i.e., the nucleotides form the so-called sequence-favorite. Compared sequences along with the sequence-favorite form the so-called working set sequences. [7]

A scoring matrix is put up where for sequences (rows in the matrix) the values of the evaluation function S are stored (by columns). For sequence (row) i in position j (column): nucleotide aij and nucleotide–favorite fij:

$$\begin{aligned} S_{ij} &= 0 \text{ in case of } a_{ij} = \text{gap} \\ S_{ij} &= 1 \text{ in case of } a_{ij} = f_{ij} \\ S_{ij} &= \text{-}1 \text{ in case of } a_{ij} \neq f_{ij} \end{aligned}$$

The elements of the scoring matrix are calculated for each sequence Sp,n+1, where n is the sequence length and p the number of aligned sequences.

The overall assessment of the quality of alignment S is calculated as:

$$S = \sum_{i=1}^{p} \sum_{j=1}^{n} S_{i,j}$$ ....Eq (1)

The extra column of the scoring matrix Sn,n+1 consists of the comparison evaluation for each sequence compared with the sequence favorite, i.e.,

$$S_{i,n+1} = \sum_{j=1}^{n} S_{i,j}$$ ....Eq (2)

The next example shows the working set. The sequence favorite is marked in red.

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

The next example illustrates the calculation of scoring matrix S. The scoring column of the counters is marked in red. It contains similarity scores for each sequence and the sequence-favorite.

1	-1	1	1	-1	1	1	3
1	1	1	1	1	-1	1	6
1	1	0	-1	1	1	-1	2

The scouts record the scores of the working sets in the list of working solutions that is organized in descending order by the total alignment scores. For each sequences (row) dynamic data structure containing the indexes of the gaps in the sequence is created. [6]

The onlookers choose the best working set and execute a local search; they make slight alterations in the working set and calculate the quality of the revised alignment.

In case of quality improvement, the alteration is accepted and the working set is stored in the list of "best temporary solutions.” [7] Otherwise, the new alignment is discarded.

The proposed approach below is for modification of the aligned working set of sequences. The column with counters of the scoring matrix S is revised and the row (sequence) with the lowest counter value (sequence that differs most from the favorite) selected. Using a random generator an index for inserting a gap (INS) and an index for deleting a gap (DEL) are selected from the list of empty positions (to keep unchanged the length of the sequence) ensuring that INS ≠ DEL. Both indexes are compared. If DEL>INS, then all characters in positions between INS and DEL are shifted one position to the right (shift right), otherwise if DEL>INS they are shifted one position to the left (shift right).

An instance for the sequence is shown below:

```
A-TGC-GGTA-CCGT-G
```

The list of gaps indexes is: 1, 5, 10, 15.

Based on a random generator, the position for insertion INS=4 and the position for deletion DEL=15 are selected. In this case, DEL>INS, the operation is shift right.

```
A-TGC-GGTA-CCGT-G
A-TG-C-GGTA-CCGTG
```

The value of similarity counter for the sequence compared with the sequence-favorite is calculated when the modification of the working set is accepted and is stored in the list of temporary best solutions; otherwise, the modification is ignored.

In case of inserting position INS=12, and deleting position DEL=5, DEL<INS and the operation is shift_left.

```
A-TGC-GGTA-CCGT-G
A-TGCGGTA-C-CGT-G
```

After a number of modifications, employed bees suspend the processing of the current working set and write down the best solution in the list of elite solutions that is sorted in descending order. The condition for termination of the parallel algorithm is the number of iterations. Then, the mother bee shall inform the queen bee of the colony and sent her the quality of the best solution (elite solution).

3.3 Hybrid Artificial Bee Colony Algorithm

3.3.1 Introduction on Differential Evolution (DE) Algorithm

Differential Evolution Algorithm is a parallel stochastic search technique designed within the common framework of Evolutionary Algorithm (EA). This algorithm is a population-based optimizer that works utilizing the concepts borrowed from EAs. The algorithm starts by sampling the search space at multiple, randomly selected search points. Like other EAs, DE creates new search points through perturbations of the existing points. Using a differential mutation and a recombination operation DE creates new search points, which are evaluated against their parents. Then a knockout selection mechanism is applied that deterministically promotes the winners to the next generation. In addition, this cycle of perturbation and selection is iterated generation after generation until the termination criteria is satisfied. [12]

3.3.2 How Hybrid ABC Algorithm works

Hybrid Artificial Bee Colony Algorithm is juxtaposed from the hybridization of the algorithms, Artificial Bee Colony Algorithm and Differential Evolution Algorithm. It could be more specified as an extension of Artificial Bee Colony Algorithm. The crossover and mutation function of Differential Evolution is implemented in Artificial Bee Colony Algorithm to get an optimized output.

DE works with a population of individuals x^i_G, $i = 1, 2, \cdot\cdot\cdot, P$ each representing a solution to the problem. DE individuals are encoded as real vectors of size N, which is the dimension of the problem. The number of individuals in a population is called population size and is denoted by P and the generation number is denoted by G. The initial population, $\mathcal{P}1$ is created by randomly creating the vectors in appropriate search ranges. Then the fitness score of each individual is calculated through evaluation. DE practices random parent selection regardless of their fitness values. In every generation, each individual $\boldsymbol{x}^{i}{}_{G}$ gets a chance to become the principal parent and to breed its own offspring mating with other randomly chosen auxiliary parents. Formally, for every principal parent $\boldsymbol{x}^{i}{}_{G}$, $i = 1, 2, \cdot\cdot\cdot, P$, three other auxiliary parents $\boldsymbol{x}^{r1}{}_{G}$, $\boldsymbol{x}^{r2}{}_{G}$, $\boldsymbol{x}^{r3}{}_{G}$ are selected randomly such that $r1, r2, r3 \in \{1, 2, \cdot\cdot\cdot, P\}$ and $i \neq r1 \neq r2 \neq r3$. Then these three auxiliary parents participate in differential mutation operation to create a mutated individual $\boldsymbol{x}^{mut}{}_{G}$ as follows:

$$\boldsymbol{x}_G^{mut} = \boldsymbol{x}_G^{r1} + F(\boldsymbol{x}_G^{r2} - \boldsymbol{x}_G^{r3}) \qquad (1)$$

Where, F is the amplification factor, a real-valued control parameter chosen from [0.1, 1.0]. Subsequently, the mutated vector, $\boldsymbol{x}^{mut}{}_{G}$, participates in exponential crossover operation with the principal parent $\boldsymbol{x}^{i}{}_{G}$ to generate the trial individual or offspring $\boldsymbol{x}^{child}{}_{G}$. Exponential crossover is actually a cyclic two-point crossover in which CR, another control parameter of DE, determines how many consecutive genes of the mutated vector, $\boldsymbol{x}^{mut}{}_{G}$, on average are copied to the offspring $\boldsymbol{x}^{child}{}_{G}$. The selection scheme used in DE is also known as knockout competition. As the name suggests, DE plays a one-to-one competition between the principal parent, $\boldsymbol{x}^{i}{}_{G}$, and its offspring $\boldsymbol{x}^{child}{}_{G}$ to select the survivor for the next generation. The DE selection scheme can be described as follows:

$$\boldsymbol{x}_{G+1}^{i} = \begin{cases} \boldsymbol{x}_G^{child} & \text{if } f(\boldsymbol{x}_G^{child}) \text{ is better than } f(\boldsymbol{x}_G^{i}) \\ \boldsymbol{x}_G^{i} & \text{otherwise} \end{cases} \qquad (2)$$

Repeating the above-mentioned mutation and crossover operations on each individual of the current generation, DE creates a new generation of population, which replaces the current generation. In addition, this generation alternation process is iterated until the termination criteria are satisfied. The control parameters of DE (F, CR and P) are chosen beforehand and are kept constant throughout the search in this canonical version of the algorithm. The pseudo-code description of canonical DE is presented in Algorithm 1.

Algorithm 1 DE

```
1: Select $P$, $F$ and $CR$ and Set $G = 1$
2: $\mathcal{P}_G$ = initialize population randomly
3: while termination criteria not satisfied do
4:     for each individual $\boldsymbol{x}_G^i$ in $\mathcal{P}_G$ do
5:         Select auxiliary parents $\boldsymbol{x}_G^{r1}$, $\boldsymbol{x}_G^{r2}$ and $\boldsymbol{x}_G^{r3}$
6:         Create offspring $\boldsymbol{x}_G^{child}$ using mutation and crossover
7:         $\mathcal{P}_{G+1}$ = $\mathcal{P}_{G+1} \cup$ Best($\boldsymbol{x}_G^{child}$,$\boldsymbol{x}_G^i$)
8:     end for
9:     Set $G = G + 1$
10: end while
```

- **jDE**

jDE algorithm is proposed that uses a self-adaptive mechanism to adjust the control parameters F and CR during the evolutionary process. In order to encode these two control parameters in individual vectors, two extends the size of each individual, $\boldsymbol{x}^i{}_G$. In other words, each individual has its own copies of F and CR, which are encoded in it and evolved with it. Therefore, the size of each individual in jDE algorithm is (N +2) where N is the size of the objective vector. For evolving new generation of individuals, the original DE mutation and crossover operations are applied. However, these genetic operations work only on the objective vector part of the individual. The F and CR for new offspring are calculated as follows:

$$F^i_{G+1} = \begin{cases} F_l + rand_1 * F_u, & \text{if } rand_2 < \tau_1, \\ F^i_G & \text{otherwise} \end{cases} \quad (3)$$

$$CR^i_{G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR^i_G & \text{otherwise} \end{cases} \quad (4)$$

where $randj$, $j \in \{1, 2, 3, 4\}$, are uniform pseudo-random values $\in [0, 1]$, τi and $\tau 2$ are constants that represent the probabilities to adjust F and CR respectively. F_l represents the minimum value that can be assigned to $F^i{}_G +1$ and the F_u represents the maximum random variation added to F_l. It should be noted that $F^i{}_{G+1}$ and $CR^i{}_{G+1}$ are obtained before the mutation is performed. Therefore, they influence the mutation, crossover, and selection operations of the new individual $\boldsymbol{x}^{child}{}_G$. In this study, $F_l = 0.1$, $F_u = 0.2$ and $\tau 1 = \tau 2 = 0.1$ were used as suggested in the original proposal. Therefore, there are two differences between jDE and the canonical DE. The first one is the realization of the F and CR parameters at individual level and the other one is the introduction of additional rules for calculating F and CR at individual level. [12]

3.4 NSGA Algorithm

NSGA Algorithm is juxtaposed from the hybridization of the algorithms, Artificial Bee Colony Algorithm and Differential Evolution Algorithm. It could be more specified as an extension of Artificial Bee Colony Algorithm. The crossover and mutation function of NSGA is implemented in Artificial Bee Colony Algorithm to get an optimized output.

- **Simulated Binary Crossover**

  Simulated binary crossover simulates the binary crossover observed in nature and is give as below,

$$c_{1,k} = \frac{1}{2}[(1-\beta_k)p_{1,k} + (1+\beta_k)p_{2,k}]$$
$$c_{2,k} = \frac{1}{2}[(1+\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}]$$

where $c_{i,k}$ is the ith child with kth component, $p_{i,k}$ is the selected parent

$\beta_k(>=0)$ is a sample from a random number generated having the density

$$p(\beta) = \frac{1}{2}(\eta_c + 1)\beta^{\eta_c}, \quad \text{if } 0 \leq \beta \leq 1$$
$$p(\beta) = \frac{1}{2}(\eta_c + 1)\frac{1}{\beta^{\eta_c+2}}, \quad \text{if } \beta > 1.$$

This distribution can be obtained from a uniformly sampled random number u between (0, 1). η_c is the distribution index for crossover. Crossover determine how well spread the children will be from their parents. [15] That is,

$$\beta(u) = (2u)^{\frac{1}{(\eta+1)}}$$
$$\beta(u) = \frac{1}{[2(1-u)]^{\frac{1}{(\eta+1)}}}$$

- **Polynomial Mutation**

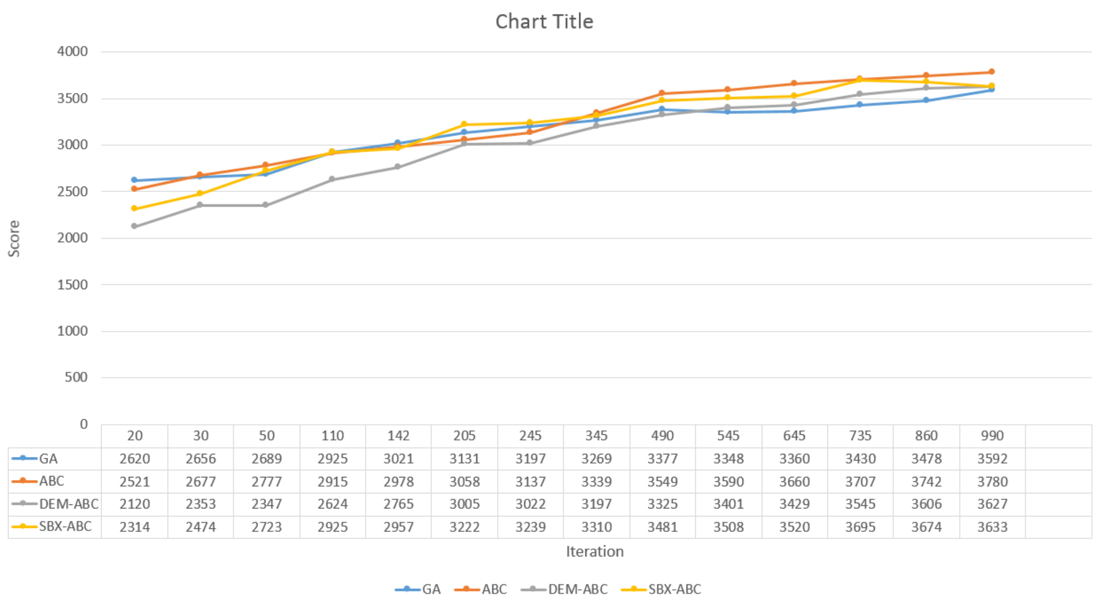
$$c_k = p_k + (p_k^u - p_k^l)\delta_k$$

where c_k is the child and p_k is the parent with p^u_k being the upper bound on the parent component, p^l_k is the lower bound and δ_k is small variation which is calculated from a polynomial distribution by using

$$\delta_k = (2r_k)^{\frac{1}{\eta_m + 1}} - 1, \quad \text{if } r_k < 0.5$$
$$\delta_k = 1 - [2(1-r_k)]^{\frac{1}{\eta_m + 1}} \quad \text{if } r_k \geq 0.5$$

r_k is an uniformly sampled random number between (0, 1) and η_m is mutation distribution index. [15]

<u>Part 4: Graphical Representation of the Evolutionary Algorithms</u>



	20	30	50	110	142	205	245	345	490	545	645	735	860	990
GA	2620	2656	2689	2925	3021	3131	3197	3269	3377	3348	3360	3430	3478	3592
ABC	2521	2677	2777	2915	2978	3058	3137	3339	3549	3590	3660	3707	3742	3780
DEM-ABC	2120	2353	2347	2624	2765	3005	3022	3197	3325	3401	3429	3545	3606	3627
SBX-ABC	2314	2474	2723	2925	2957	3222	3239	3310	3481	3508	3520	3695	3674	3633

The graphical representation here shows the scores against iteration graph for the four evolutionary algorithms. All the four algorithms here show very competitive performance in solving the Multiple Sequence Alignment problem. Time complexity may vary from computer to computer due to computer configuration. However, their fitness value is very accurate. From all the works of the implemented algorithms, it is seen that Artificial Bee Colony Algorithm is much optimized in terms of its score.

Part 5: Conclusions

In conclusion, after implementing all the four evolutionary algorithms, it is seen that Artificial Bee Colony Algorithm shows a better performance in terms of its scores compared to the Hybrid Artificial Bee Colony Algorithm, NSGA Algorithm and Genetic Algorithm.

The proposed algorithms behave very efficiently in solving the multiple sequence alignment problem. They are very competitive in solving this problem. However, ABC stands ahead of all the other algorithms due to its robustness and optimized iterative ability compared to the rest of the algorithms.

On a separate note, the hybrid algorithms performed better than the evolutionary Genetic Algorithm at many iteration in the graphs. These constructive and composed algorithms, which are implemented on Artificial Bee Colony algorithm, proved its efficiency than the well-known Genetic Algorithm even.

Part 6: Reference

[1] Cédric Notredam Desmond G. Higgins. "SAGA: sequence alignment by genetic algorithm", EMBL outstation, The European Bioinformatics Institute, Hinxton Hall, Hinxton, Cambridge CB10 1RQ, UK, December 5, 1995.

[2] http://link.springer.com/article/10.1007/s10489-009-0189-4#page-1

[3] Yixin Chen, Yi Pan, Ling Chen, Juan Chen." Partitioned optimization algorithms for multiple sequence alignment."

[4] https://herstat.com/blog/23-can-bees-communicate-the-location-of-propolis-by-dancing.html

[5] Kosmas Karadimitriou, Donald H. Kraft." GENETIC ALGORITHMS AND THE MULTIPLE SEQUENCE ALIGNMENT PROBLEM IN BIOLOGY". Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, Baton Rouge, LA, February 1996.

[6] Plamenka Borovska, Veska Gancheva. "Massively Parallel Algorithm for Multiple Sequence Alignment Based on Artificial Bee Colony"

[7] Xiujuan Lei, Jingjing Sun, Xiaojun Xu, Ling Guo. "Artificial Bee Colony Algorithm for Solving Multiple Sequence Alignment"

[8] Seeley T.D., The Wisdom of the Hive, Harvard University Press, Cambridge, MA, 1995.

[9] http://www.incogen.com/bioinfo_tutorials/Bioinfo-Lecture_4-multiple-sequence-align.html

[10] http://en.wikipedia.org/wiki/List_of_sequence_alignment_software

[11] http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html

[12] Nasimul Noman, Danushka Bollegala, Hitoshi Iba. "An Adaptive Differential Evolution Algorithm."

[13] Artificial Bee Colony (ABC) Algorithm http://mf.erciyes.edu.tr/abc/

[14] Genetic Algorithm http://www.cs.umb.edu/

[15] Aravind Seshadri. "A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II"

[16] FTP NCBI GENBANK ftp://ftp.ncbi.nlm.nih.gov/genbank/docs/