Solving Fragment Assembly Problem using hybrid 2-Stage Algorithm

Aziz Mohammad(011 143 065)
Md. Al-Hasan(011 142 062)
Md. Arifur Rahman(011 143 053)
Md. Shakhaeat Hossain(011 143 034)

Department of Computer Science and Engineering
United International University

A thesis submitted for the degree of
BSc in Computer Science & Engineering

July 2019

Declaration

We, Aziz Mohammad , Md. Al-Hasan , Md. Arifur Rahman and Md. Shakheat Hossain, declare that this thesis titled, Solving Fragment Assembly Problem using hybrid 2-Stage Algorithm and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for a BSc degree at United International University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.
- Where the thesis is based on work done by ourselves jointly with others, we have made clear exactly what was done by others and what we have contributed ourselves.

Signed: _____

Date: _____

(Student Name)

Certificate

I do hereby declare that the research works embodied in this thesis entitled "Solving Fragment Assembly Problem using hybrid 2-Stage Algorithm" is the outcome of an original work carried out by Aziz Mohammad , Md. Al-Hasan , Md. Arifur Rahman and Md. Shakheat Hossain under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of BSc in Computer Science and Engineering.

Signed: _____

Date: _____

(Name of the Superviser)

Department of Computer Science and Engineering,

United International University,

Dhaka-1209, Bangladesh.

Abstract

Deoxyribonucleic acid (DNA) is the carrier of the hereditary characteristics of an organism. It also holds promise for predicting fatal diseases. These predictions are essential as they enable prevention rather than cure. However, such detection techniques would be improbable without comprehending the complete genome of an organism. Through our work we tried to unravel the genome code by assembling DNA fragments using our 2-stage hybrid algorithm which employs Ant Colony Optimization and deft Local Search technique to maximize the DNA fragment overlap count.

In memory of our families and teachers who dared to believe in us.

Acknowledgements

Please write the acknowledgement here.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Deoxyribonucleic acid (DNA) is the carrier of the genetic information, a self-replicating material present in nearly all living creature considers the main substance of chromosomes. 29 years ago in 1990 the Human Genome Project initiated with the objective of determining the exact DNA sequence of the entire euchromatic human genome generally known as nucleotide base pairs within 15 years [3]. The DNA fragment assembly is a technique that was brought forward to reconstruct this DNA sequence from a large number of DNA fragments [4] that are shorter than 1000 bases as current technologies usually sequence DNA fragments shorter than 1000 bases[5], where human genome contains about 3.2 billion nucleotide base pairs[1] . This paper is going to discuss about a Fragment Assemble Problem (FAP) solving idea using hybrid 2-Stage Algorithm for sequencing the DNA.

1.1 Motivation

Having the essentially complete sequence of the genome is similar to having all the pages of a manual embodied with in an organism. The DNA FAP has great contribution in the genome project as it helps to explore the DNA sequence information which is vital for medical, agricultural, gel electrophoresis [6] and many other research areas. The fragment assembly problem (FAP) of DNA sequencing has the potentiality to disclose a bulk amount of information for the different types of diseases carrier like autosomal recessive disorders, genetic risk factors for complex adult-onset diseases and other predictive medical and non-medical information which may enable a new door step for research into the basis of genetic disease.

1.1.1 Elementary Backdrop

In modern times, biological science illustrate a clear portrait of DNA the fundamental elixir of all the living beings. The most basic code of functionalities for all organisms are being stored in this Deoxyribonucleic Acid (DNA). Which is basically composed of four types of molecules called nucleotides or bases such as Adenine (A), Thymine (T),

Cytosine (C), and Guanine (G). The hereditary information of an organism encoded through the sequences of these four nucleotides with in a DNA. With this information, we can easily decode the basic function and malfunction of the organism. We can flourish other fields of science like as agriculture, medical, microbiology, pharmaceuticals and other researches areas which are dependent on the biological advancement.

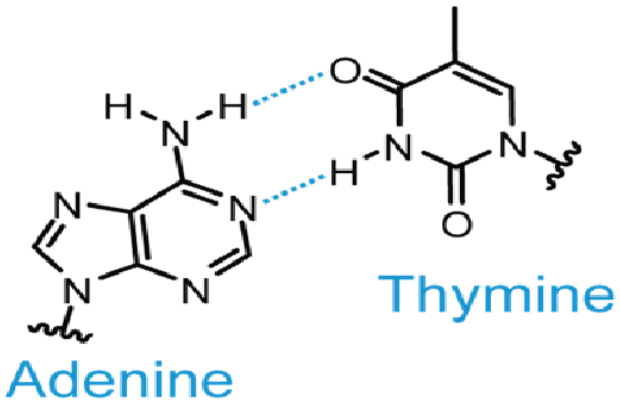

Figure 1.1: Chemical structure of DNA - hydrogen bonds shown as dotted lines

The information extract from the DNA sequence is very vital for the researches. But there are so many difficulties and crucial reality to do this. Determining the protein synthesis of an organism, actually knowing the DNA sequences are very important for scientists to understand their subject of research. How they are really functioning, the history of their heredity is very vital to reach a decision.

At the beginning it was not so easy to sequence the DNA. Initially 3.5 billion base pairs of human genome sequencing was accomplished in 2001 [1] by the shotgun sequencing method after the Frederick Sanger et al. in 1982 [1]. Advancement in DNA fragment assembly algorithms has momentous contribution for the success of the shotgun sequencing method. But all the FAP algorithms has to pass over the sequencing errors 1.2, unknown orientation 1.3, incomplete coverage 1.4, and repeated regions 1.5 challenges [1]. Efficient and computationally accurate algorithm can less the expense, time and intervention of biologists for the DNA fragment assembly. Better algorithm can improve the efficiency of DNA fragment assembly more easily. Dual helix structure of DNA, where fragment can come from each strand of the helix structure made it more difficult to accomplished, Hence, it is a NP-hard problem with $2^k * k!$ combinations, assuming k is the given fragments [7].

Figure 1.2: Part of DNA sequence – prototypification of complete genome of virus

1.1.2 DNA Fragment Assembly Problem

DNA fragment assembly problem has been appeared to read or assembling the fragments of contigs. Each contigs have several overlapping fragments of DNA. In DNA sequencing, the challenge is to reconstruct the contigs accurately. But this one is one of the important and complex task of this process. Such way DNA fragment assembly problem came in front of us and became one of the crucial challenges for the computational biologists like brain teasing problem in real life. Practically even more rough to accomplish. The DNA FAP is considering the last step in genome project. [7]. In this process someone has to work with a given bulk amount of DNA fragments to read, perhaps an erroneous one. Here, the challenge is to find the correct sequence of the DNA on the context of permutation of given fragments which represents the original DNA sequence in a much better way. Still there is so many limitation of current technology for this. Thus, this filed is still wide open to contribute properly to meet the challenge of scaling up this to real one. Hence, fragment assembly problem solving algorithm has been introduced on behalf of this.

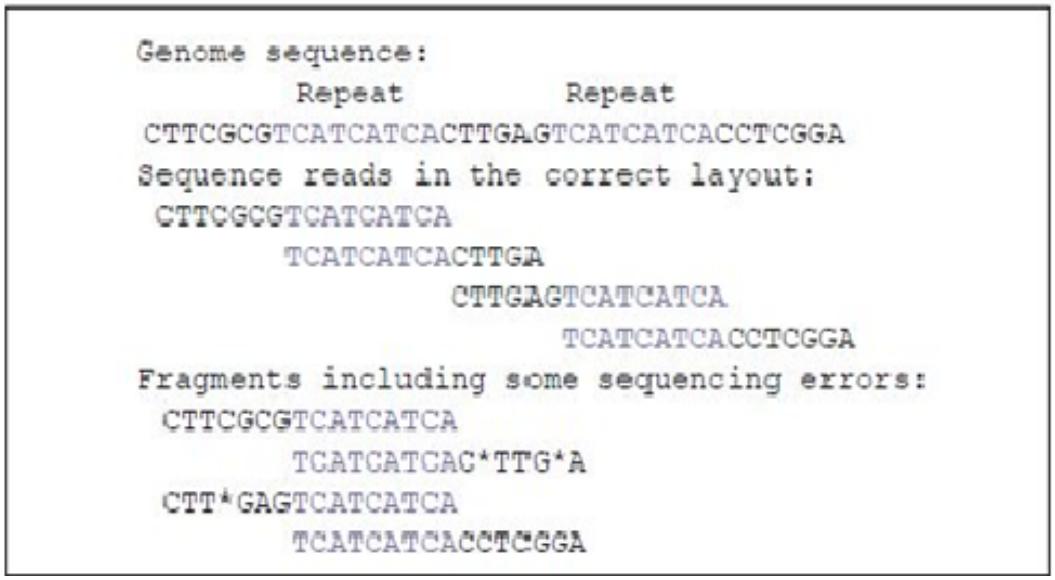

Figure 1.3: Assembly errors caused by fragments errors [1]

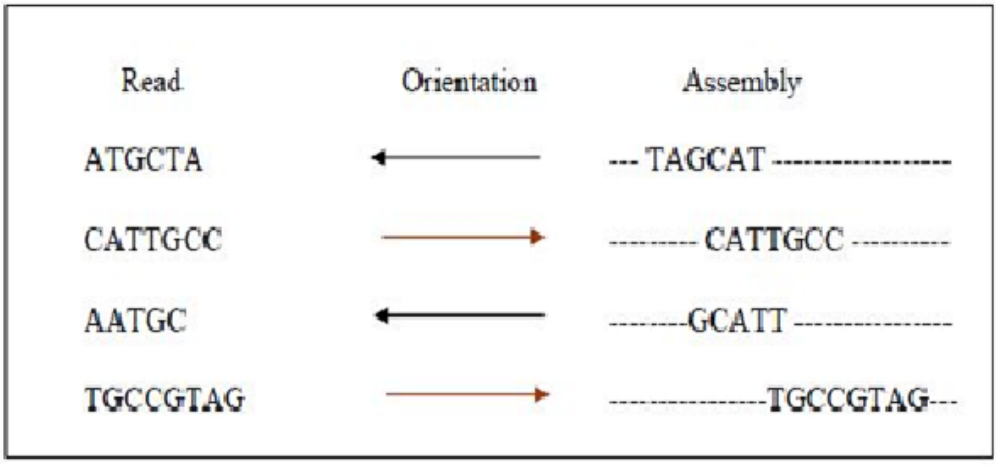

Figure 1.4: Calculating reverse complement of DNA fragments [1]

1.2 Objectives of the Thesis

Our objective of this thesis is to find a better hybrid algorithm which will give a much better result of DNA fragment assemble problem from the existing ones.

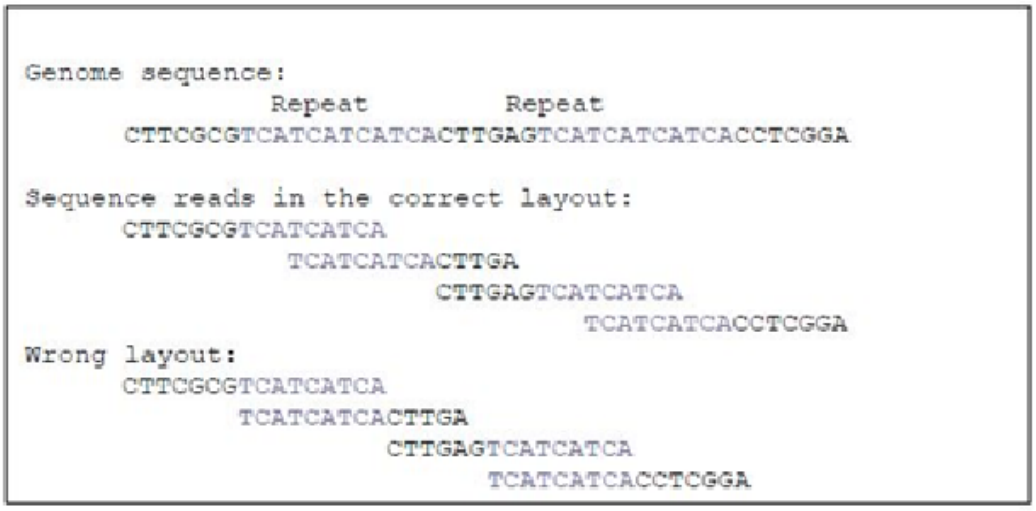

Figure 1.5: Assemble errors caused by repeats

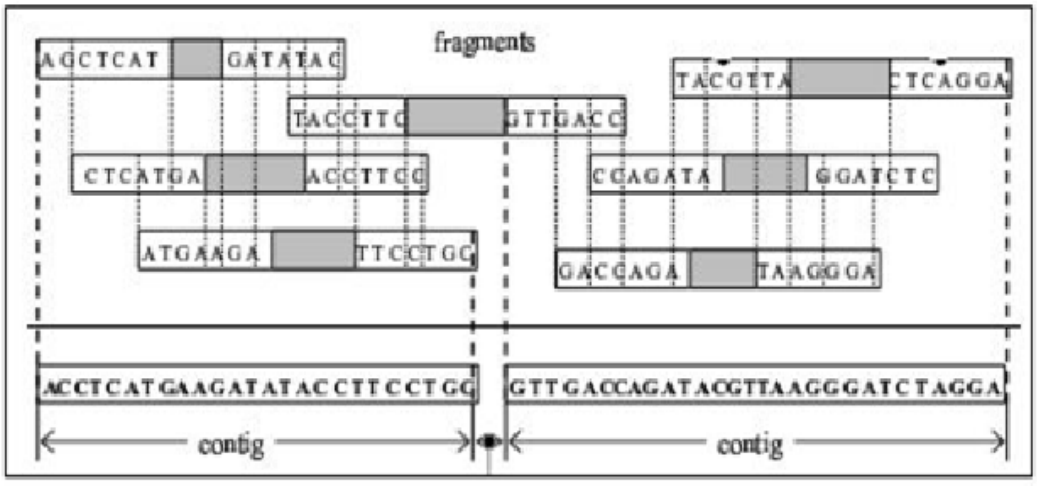

Figure 1.6: Unable to rebuild the objective sequence due to no fragment coverage

1.3 Thesis Contributions

In this thesis, we focus on assessing the performance of hybrid 2-stage algorithm for solving the DNA fragment assembly problem with some bench mark dataset. We have selected ant colony optimization and local search algorithm for this. Ant colony optimization (ACO) is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems.And local search is being used to solve computationally hard optimization problems which is a heuristic method.

1.4 Organization of the Thesis

The thesis is organised as follows:

Chapter 2 provides related works.

Chapter 3 presents the proposed method.

Chapter 4 discusses the results and experimental analysis.

Chapter 5 presents the conclusions, summaries the thesis contributions, and discusses the future works.

Chapter 2

Related Work

The gravity of determining the complete genome sequence and its genetic functionality made the DNA fragment assembly problem one of the most vital subject of bioinformatics technology. As an interdisciplinary field of science, bioinformatics combines biology, computer science, information engineering, mathematics and statistics to analyze and interpret biological data. Here, we are going to discuss about the ground and development process of the DNA fragment assembly problem as well as our focusing point of this thesis.

2.1 Genome project and DNA sequencing

The Genome projects are designed upon an especial aim to determine the complete genome sequence, annotate protein-coding genes and other important encoded genome stuffs of an organism either it be an animal, archaean, a plant, fungus, bacterium, protest or a virus. Biologically all these species have one to several chromosome. Basically, Genome projects are scientific effort to explore and extract genome sequence (DNA sequences) of these chromosomes. Thus, Human Genome project was a significant output of it. Today, it has a great contribution in life sciences and its developments. Here is a time line of evaluation of this project to understand its historical phage.

2.1.1 Scope of DNA fragment assembly in genome project

Existing technologies is unable to read the whole DNA sequence of an organism at a time. It can read small pieces of genome sequence between 20 to 30000 bases depends on technology it used. In shotgun sequencing technique the whole DNA of a living being is first crush into millions of small pieces. Which are called DNA fragments. After that these DNA fragments go under a complex computational process to assemble in order to reconstruct the original sequence. To do this, these DNA fragments are read by an automated sequencing machine which can read up to 1000 nucleotides of DNA at time.

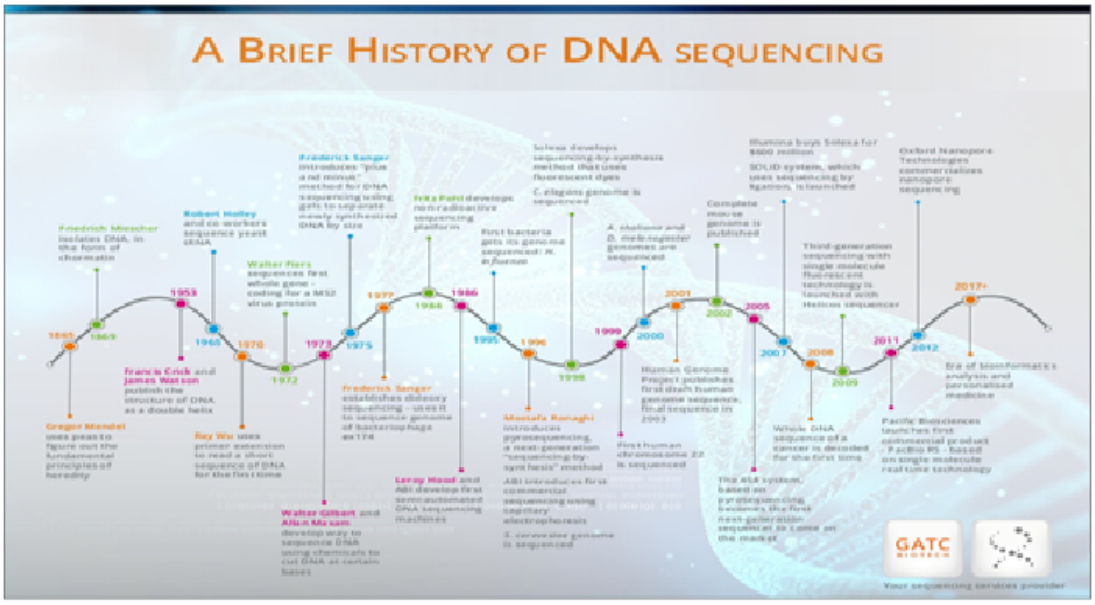

Figure 2.1: A brief history of DNA sequencing

But the realities of genomes and large number of similar sequence known as repeats converted this task to a difficult problem to solve.

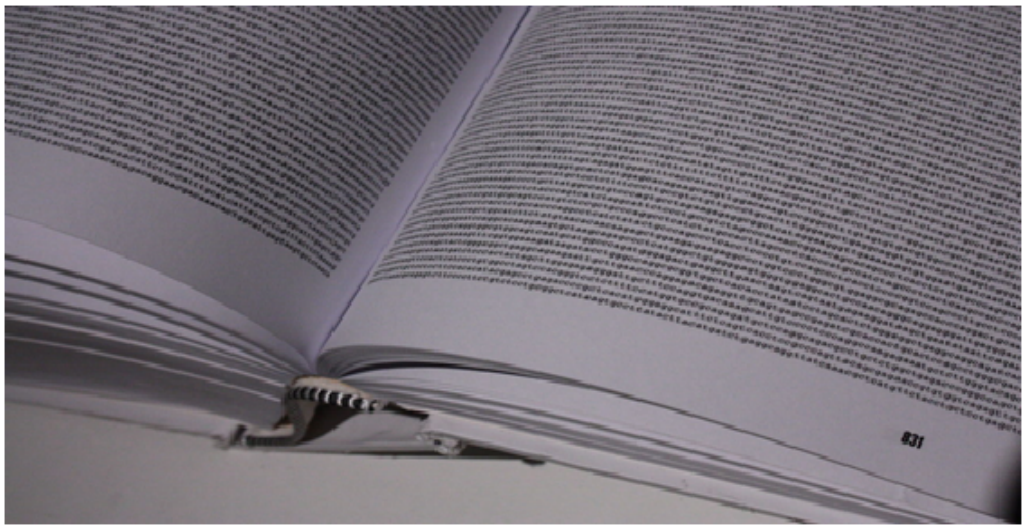

Figure 2.2: printed human genome sequence- need more than 100 huge books

2.1.2 Problem breakdown

In sequence process 3.2 billion nucleotides of human DNA cannot be read at once thus it cut at random sites to make fragments as stated above. Then all the DNA fragments are trying to assemble through an automated sequencing process algorithm to reconstruct the whole DNA again. This assembling processes have to go beyond three distinct phases known as OLC in shotgun sequencing technique and was very common with the Sanger-data (Miller et al, 2010) [8].To understand this problem we have to understand the following factors of DNA sequencing more properly:

Fragment of DNA : Fragments are basically the genome sequence can be long up to 1000bps in length, randomly cut from a DNA which is a double helix of two anti-parallel and fulfilling nucleotides sequences [2012]. The sequences are built in pairs.In forming sequence Adenine (A) always do pair with Thymine (T) and Guanine (G) with Cytosine (C). There exist two strand where one is read form 5' to 3' and another one is 3' to 5'.

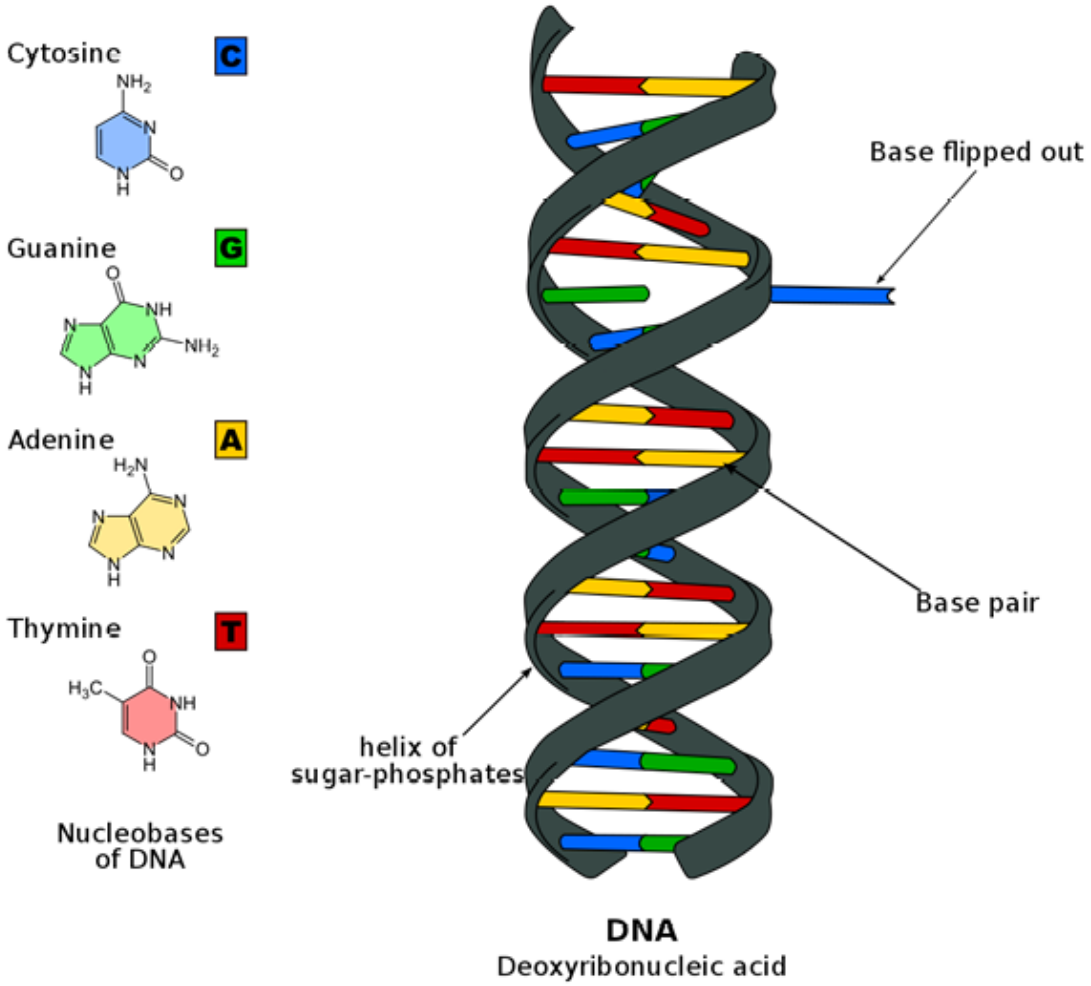

Figure 2.3: DNA double helix structure

Prefix : A sub-string comprising the first N characters of a fragment.

suffix : A sub-string comprising the last N characters of a fragment.

Here, the value of N depends on the overlap. If the common sequence between the suffix of fragment x and prefix of fragment y extend up to 2 characters then the value of the N will be 2 or if it is then the value on N will be 3 and so on.

Contig : The word "contig" is used in DNA sequence to illustrate a set of contiguous (adjacent) overlapping fragments. Where, the overlap between adjacent fragments is greater than a predefined orbit.

Overlap: Resemble the common sequence between the suffix and prefix of fragments.

Unknown orientation: When the DNA cut at random sites to make fragments, the original orientation of its sequence is lost. At the time of assembling it can be read in either 3' to 5' or 5' to 3'. But how can one decide which strand have to choose? It's very difficult because if one does not find any overlap at first attempt, it may possible to find overlap within their reverse complement.

Base call errors: In the electrophoresis procedure, experimental faults create three types of errors known as base call errors. They are substitution, insertion, and deletion errors. These errors affect the overlap detection process among fragments. Further, the impact goes on setting alignments for consensus process.

Incomplete coverage: This happens when an algorithm can not assemble a set of DNA fragments into a contig.

Repeated regions: DNA sequencing projects face a problem of repetition caused by "repeated region" of the selected DNA. It may appear two or more times in that process. Still any assembly programs cannot handle it exactly.

Chimeras: It happens when two fragments which are not contiguous in the target DNA molecule join together into one fragment.

Contamination: If the fragment from the vector DNA remains impure then the contamination occurs.

In genome project DNA fragment assembly is the last step of DNA sequencing. It will be better to say the last challenge rather than last step because passing over the three distinct phase of shotgun sequencing are not as easy as it seems to be. Say:

Overlap Phase(O): The challenge of this phase is to find the best overlapped match between the pairs of fragment. In that process the prefix of a fragment and the suffix of another one are compared through a dynamic algorithm to construct a semi global alignment of fragments.

Layout phase (L): The most difficult phase of DNA fragment assembly is the layout phase because at the time of shotgun sequencing process the layout of DNA fragment cannot be preserved. Especially in which order and from which DNA strand the fragments come it's really hard to determine. The challenge is here to finding the order of fragments on the computed similarity scores where search space for an optimal layout is $2\hat{k}*k!$, here k is the number of fragments, $2\hat{k}$ is all the combinations of fragments' orientations and k! represents all the permutations of fragments . But there are some other issues which complicated this step even more to conduct known as unknown orientation, base call errors, incomplete coverage, repeated regions, chimeras and contamination as stated above. Thus , This step is proven to be an NP-hard problem Pevzner (2001) [9]. At the end of this step a progressive alignment algorithm is applied to combine all the pair wise alignments after having an optimal order.

Consensus Phase (C): The objective of this phase is to reconstruct the original DNA sequence on the basis of majority vote rule from the optimal layout of fragments obtained in the layout phase. To check the accuracy of consensus anyone can use the law of distribution of coverage [10]

$$Coverage = \frac{\sum_{i=0}^{n} \text{length of the fragment } i}{\text{target sequqnce length}} \tag{2.1}$$

Illustrating OLC phases [10] through an example

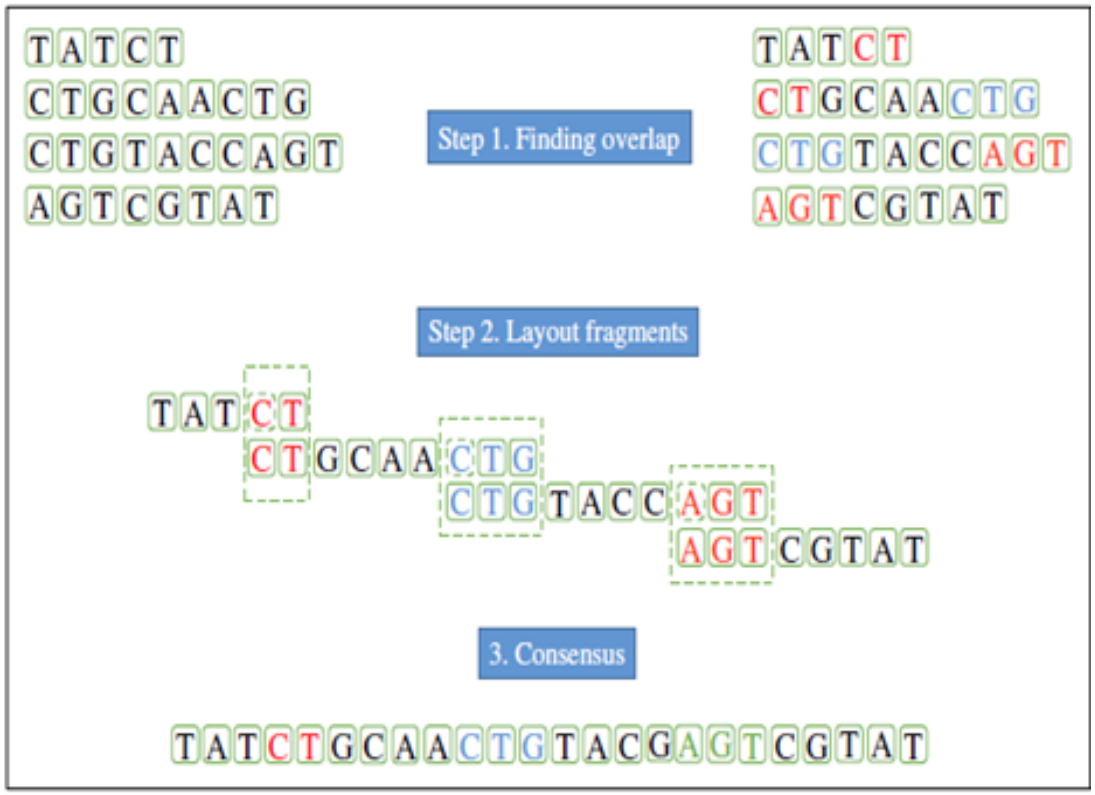

Figure 2.4: example of OLC phases through an example

Here, we considering four DNA fragments F1= TATCT, F2=CTGCAACTG, F3=CTGTACCAGT and F4=AGTCGTAT. At the overlap phase we determined the overlap among these fragments comparing the prefix of one fragment and the suffix of another one. Then we construct the possible optimal layout on computed similarity scores to find the original order and orientation of the fragments. After that in consensus phase we reconstruct the original DNA sequence TATCTGCAACTGTACCAGTCGTAT upon majority vote rule.

2.1.3 Initiatives to solve this problem

In quest of solving DNA fragment assembly problem many algorithms have been developed. As this problem is NP-hard one so the exact solution of this problem is very hard to find. Therefore many heuristic and meta-heuristic disciplines have been applied on this problem to find a best one. Genetic algorithm, pals algorithm and poems algorithm are some of them.

Genetic Algorithm: [11] heuristically tuned genetic algorithm to solve DNA fragment assembly problem. The authors introduce two ideas (1) Chromosome Reduction Step (CRed) and (2) Chromosome Refinement Step (CRef) to improve the efficiency of Genetic algorithm in their research. They used a chromosome

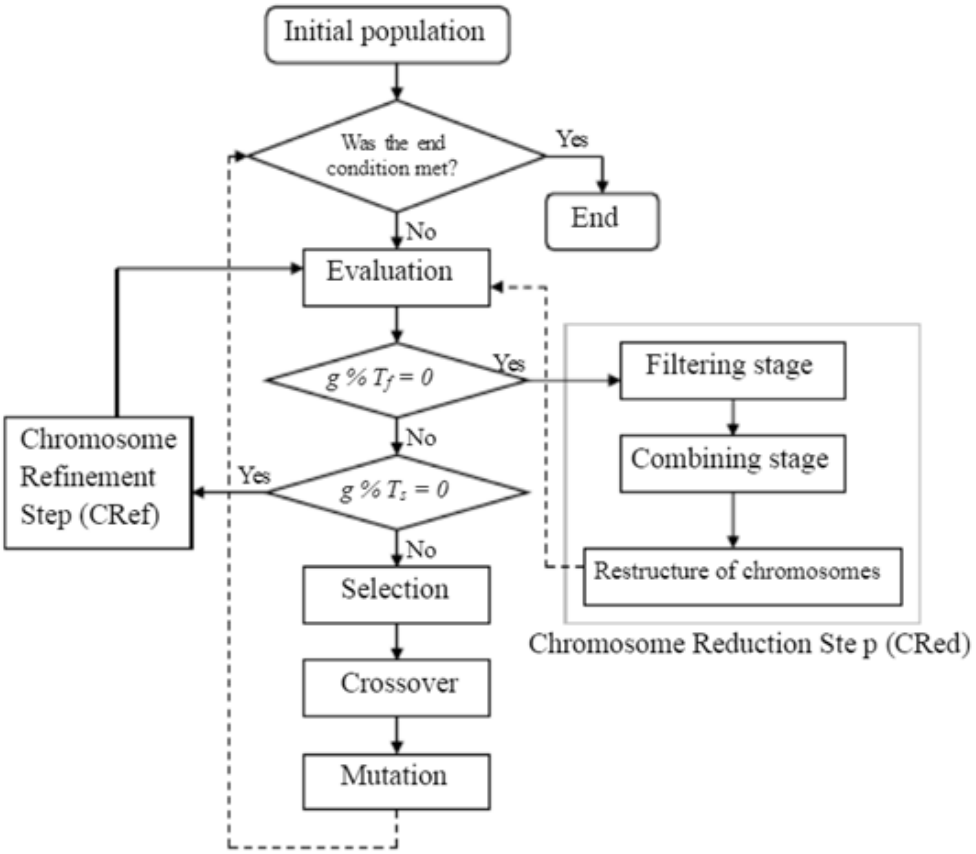

Figure 2.5: flowchart of CRed and CRef applied on genetic algorithm

to describe the permutation of DNA fragments. These fragments are labeled in serial 1 to N, here N is the total numbers of the fragments. And the underneath fitness function is the sum of similarity measures of all adjacent fragments.

$$Fitness(c) = \sum_{i=0}^{n-2} similarity(i, i+1) \tag{2.2}$$

Their proposed algorithm is as follows including CRed and CRef.

Problem Aware Local Search (PALS):
E. Alba and G. Luque [12] construct a new local search algorithm entitled as problem aware local search (pals) in 2007 to solve DNA FAP. Their key contribution on the DNA fragment assembly problem is to indirectly estimate the number of contigs rather than using traditional fitness function by measuring the number of contigs that are created or destroyed when tentative solutions are manipulated. They proposed a variation of Lin's 2-opt [13] which considers both the overlap of fragments and the number contigs created or destroyed at the same time. Their proposed algorithm is:

Algorithm 1 Pseudo-code for PALS algorithm

```
1: s ← GenerateInitialSolution() {Create the initial solution}
2: repeat
3:    L ← 0
4:    for i =0 to N do
5:       for j =0 to N do
6:          Δ_c, Δ_f ← CalculateDelta(s,i,j) {See Algorithm 2}
7:          if Δ_c ≥0 then
8:             L ← L∪ < i, j, Δ_f, Δ_c > {Add candidate momvement to L}
9:          end if
10:      end for
11:   end for
12:   if L <> ∅ then
13:      < i, j, Δ_f, Δ_c >← Select(L) {Select a movement among the candidates}
14:      ApplyMovement(s,i,j) {Modify the solution}
15:   end if
16: until no changes
17: return s
```

Prototype Optimization with Evolved iMprovement Steps (POEMS): The objective of POEMS algorithm is to find the best modification of the current solution known as prototype in each iteration. It is an iterative optimization approach which applies an evolutionary algorithm where modification basically represents the sequence of fixed length of primitive actions for the problem at hand. A set of action sequence are passed as an input parameter to the evolutionary algorithm EA to check the best evolved action sequence whether it affects the current prototype or not. If something better or similar modified prototype is found then it will be considered as a new prototype for the next iteration otherwise present one prevails. Jiri Kubalk , Petr Buryan and Libor Wagner [7] uses this algorithm in 2010 to solve DNA FAP.

Algorithm 2 Pseudo-code for CalculateDelta(s,i,j) function in PALS

1: $\Delta_c \leftarrow 0$
2: $\Delta_f \leftarrow 0$
{Calculate the variation in the overlap}
3: $\Delta_f = w_{s[i-1]s[j]} + w_{s[i]s[j+1]}$ {Add the overlap of the modified solution}
4: $\Delta_f = \Delta_f - w_{s[i-1]s[i]} - w_{s[j]s[j+1]}$ {Remove the overlap of the current solution}
{Test if a contig is broken, and if so, it increases the number of contigs}
5: **if** $w_{s[i-1]s[i]} > cutoff$ **then**
6: $\Delta_c = \Delta_c + 1$
7: **end if**
8: **if** $w_{s[j]s[j+1]} > cutoff$ **then**
9: $\Delta_c = \Delta_c + 1$
10: **end if**
{Test if two contig are merged, and if so, it decreases the number of contigs}
11: **if** $w_{s[i-1]s[j]} > cutoff$ **then**
12: $\Delta_c = \Delta_c - 1$
13: **end if**
14: **if** $w_{s[i]s[j+1]} > cutoff$ **then**
15: $\Delta_c = \Delta_c - 1$
16: **end if**
17: **return** Δ_f, Δ_c

There are several others algorithm which have been used in solving DNA fragment assembly problem. Such as:

Table 2.1: List of different algorithms

Algorithm Names	**Research Authors**
Bat Algorithm	Taher Muhammad Mahdee, Md. Habibur Rahman,Mo
A hybrid crow search algorithm	Mohcin Allaoui, Belaid Ahiod, Mohamed El Yafrani
A memetic gravitation search algorithm	Ko-Wei Huanga,d, Jui-Le Che et. al.
The Firefly approach	Anna Bou Ezzeddine et al.
A memetic particle swarm optimization algorithm	Ko-Wei Huang ,Jui-Le Che et. al.
Bee Algorithms	Jesun Sahariar Firoz et. al.
A Reinforcement Learning Approach	Maria-Iuliana Bocicor et. al.
Using Iterative Optimization with Evolved Hyper mutations	Ji rí Kubalík et. al.

2.2 Summary

In this chapter we have discussed about the DNA fragment assembly problem step by step. It's scope in the genome project and the applied methods to solve this problem efficiently till now. In the next chapter, we will illustrate our proposed method to decipher this problem.

Algorithm 3 Prototype Optimization with Evolved iMprovement Steps(POEMS)

1: $i \leftarrow 0$
2: **generate**($Prototype^{(i)}$)
3: **repeat**
4: $i \leftarrow i + 1$
5: $BestSequence \leftarrow$ **run_EA**($Prototype^{(i-1)}$)
6: $Candidate \leftarrow$ **apply**($BestSequence, Prototype^{i-1}$)
7: **if** $Candidate$ **not_worse_than** $Prototype^{(i-1)}$ **then**
8: $Prototype^{(i)} \leftarrow Candidate$
9: **else**
10: $Prototype^{(i)} \leftarrow Prototype^{(i-1)}$
11: **end if**
12: **until** POEMS termination condition
13: **return** $Prototype^{(i)}$

Chapter 3

Proposed Method

In this paper we have attempted to accomplish a two-stage hybrid algorithm that solves the covetous fragment assembly problem. The proposed algorithm aims to solve the aforementioned problem using ACO algorithm in the first stage and carefully maneuvers the obtained outcome using LS in the second stage to further ameliorate the result. ACS algorithm allows us to find the maximum overlapping DNA sequence using a coordinated set of agents called Ants that rely on "pheromones" to meta-heuristically find the optimum solution. The generated outcome is then fed to the Local Search algorithm which improvises it using a series of random swaps to produce better DNA sequence.

3.1 Problem Definition

Given, a set of fragments and an associated matrix containing the overlapping value between fragments. Here,$f = f_0, f_1, f_2, ..., f_n$ is a set of fragments. The overlap score between i^{th} fragment and j^{th} fragment is in p_{ij} location of the matrix.

3.2 Objective

We need to maximize the overlap score to find the best consensus sequence from $f_0, f_1, f_2, ..., f_n$. We use the fitness function in (3.1) to calculate the sum of the overlaps between fragments.

$$Overlap(L) = \sum_{i=0}^{N-1} w_{s_{[i]}, s_{[i+1]}} \tag{3.1}$$

3.3 Stage I

In this stage, we employ a set of agents, called ants, who search in parallel for maximum overlap score while formulating the problem as a TSP and cooperate with each other through pheromone-mediated indirect and global communication system.

3.3.1 Introduction to Ant Colony System

Ant colony algorithms are based on natural metaphors. Basically it imitates ant colonies where real ants search for a food source using optimized path solution from their nests [2]. Ants eject pheromone [5] to leave their mark along the path they tread for the next ant as a clue or information without using visual cues.

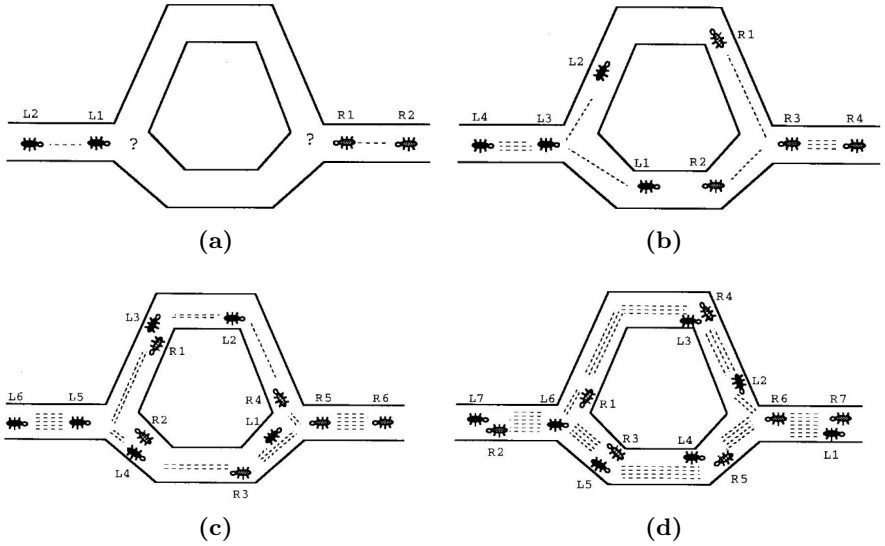

Figure 3.1: Different steps of ant traversing the graph and evaporation of pheromones [2]

In figure 3.1 we see how ant traverse towards the food resource. They eject pheromone while looking for food resources. Consider figure 3.1a: ants randomly select left or right since they have no clue whether to choose left or right. Figure 3.1b half of ant select right and rest of them selects left. On figure 3.1c because the speed of all ant are same and ants are coming from both side left and right(denoted with R as right and L as left). Instantly following the dashed line as pheromone, the amount of pheromones will be greater in the shorter path than in the longer path. More ants will take to the shorter path on average. So after a period of time the amount of pheromone will increase on the shorter path and it will increases the difference between two paths. So it will help other ants to take the decision for choosing the path for walking using the probability. On figure 3.1d Now new ants can easily choose the path on the decision point because the amount of pheromone is simultaneously increasing on the desired path with a positive effect. And very soon all the ants will follow the selected shortest path edged by lots of pheromone.

This meticulous mechanism carried out by ants in their niche lays the foundation for Ant Colony System (ACS) algorithm. ACS tries to solve the Traveling Salesman Problem (TSP) by mimicking the nature of ants. To read further about TSP please refer to Appendix **??**.

Ant system utilizes a graph representation expanded as follows: in addition to the

cost measure, each edge has also a desirability measure, called pheromone, which is updated at run time by artificial ants.

Each ant then freely traverses the graph moving from one city to another. However the ant bifurcates its decision. The ant may decide to explore a new edge or it may decide to rely on its priori and accumulated knowledge to make a calculated move.

$$s = \begin{cases} \arg\max_{u \epsilon J_k(r)} \{[\tau(r,u)].[\eta(r,u)]_\beta\}, & \text{if } q \leq q_0 \text{ (exploitation)} \\ S, & \text{otherwise (biased exploration)} \end{cases} \tag{3.2}$$

An ant in node r uses the pseudo-random-proportional-rule [2] mentioned in 3.2 to proceed to the next node. Here,

- q is a random number uniformly distributed in the range $[0 \ldots 1]$.
- q_0 is a parameter where $0 \leq q_0 \leq 1$.
- S is another random variable as in (3.3).

An ant will always prefer a shorter edge with higher amount of pheromone. However this is balanced out by the introduction of the parameter q_0. If $q \leq q_0$, then the ant opts for exploitation 3.2 else it chooses biased exploration 3.3, where q is a random number generated between 0 and 1.

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)].[\eta(r,s)]^\beta}{\sum_{u \epsilon J_k(r)} [\tau(r,s)].[\eta(r,s)]^\beta}, & \text{if } s \epsilon J_k(r) \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

In (3.3) we come across the expression used by the k^{th} ant in node r chooses to move to node s. Here τ is the pheromone in the edge, η is the inverse of the distance between node r and s. J_k is the set of all unvisited nodes by the ant. β is a parameter aimed to balance the importance of the pheromone vs. the distance.

Now that we are aware of the fact how the ant moves from one node to the other. We will quickly glance at the pheromone deposition mechanism used by the ants.

ACS makes use of two ways to update the pheromone level of the graph [2]. Namely:

- Local Updating Rule.
- Global Updating Rule.

Local Updating Rule: Each ant while making a tour through the graph deposits and changes the amount of pheromone on the edges they visit using the following rule (3.4):

$$\tau(r,s) = (1-\rho).\tau(r,s) + \rho.\Delta\tau(r,s) \tag{3.4}$$

Here ρ is a pheromone decay parameter in the range 0 and 1.$\tau(r, s)$ is the desirability measure of the edge.

Global Updating Rule: The global updating rule is very much different from the local updating rule in the sense it only allows the ant that weaved the global best tour to deposit pheromone. We make the search more directed using the above rule and a pseudo-random-proportional rule.

$$\tau(r, s) = (1 - \rho).\tau(r, s) + \rho.\Delta\tau(r, s) \tag{3.5}$$

where

$$\Delta\tau(r, s) = \begin{cases} (L_{gb}), & \text{if } (r, s)\epsilon \text{ global-best-tour} \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

Here, L_{gb} is the length of the global best tour. Since we are considering the maximum overlap count we consider the length itself instead of the inverse of it used in [2].

3.3.2 Using ACS to find longest sequence

In the previous section, we got a brief notion of ACS. In this section we will further discuss, how we applied ACS to estimate the maximum overlap sequence. The process initiates by randomly placing ants on different node of the graph, while making sure no two ants occupy the same node. Now the number of ants used for traversal have been explained in Section **??**. Our main motive in this step is to generate enough tours for Stage II of the proposed algorithm. So after careful consideration, we considered maximum number of tours be 100. Another important decision that we were faced with is when the entire process should halt. We discuss number of iterations in Section **??**.

As we can see from the flowchart, the process proceeds as the ants move across the nodes one by one using (3.2) while updating the pheromone of the edges using local updating rule in (3.4).

Once the ants have finished their tours, we apply the global updating rule from (3.5). This allows the ant with best tour to update the pheromone level of the path taken by it, thus, making the search more directed towards a globally best solution.

At the end of this process, we receive a set of gene fragment sequences. This final set undergoes further analysis in the second stage of the algorithm to beget even better overlap counts.

3.4 Stage II

In this stage, we aim to further increase the overlap count using simple non-deterministic local search technique on the set of best tours obtained from Stage I of the proposed method.

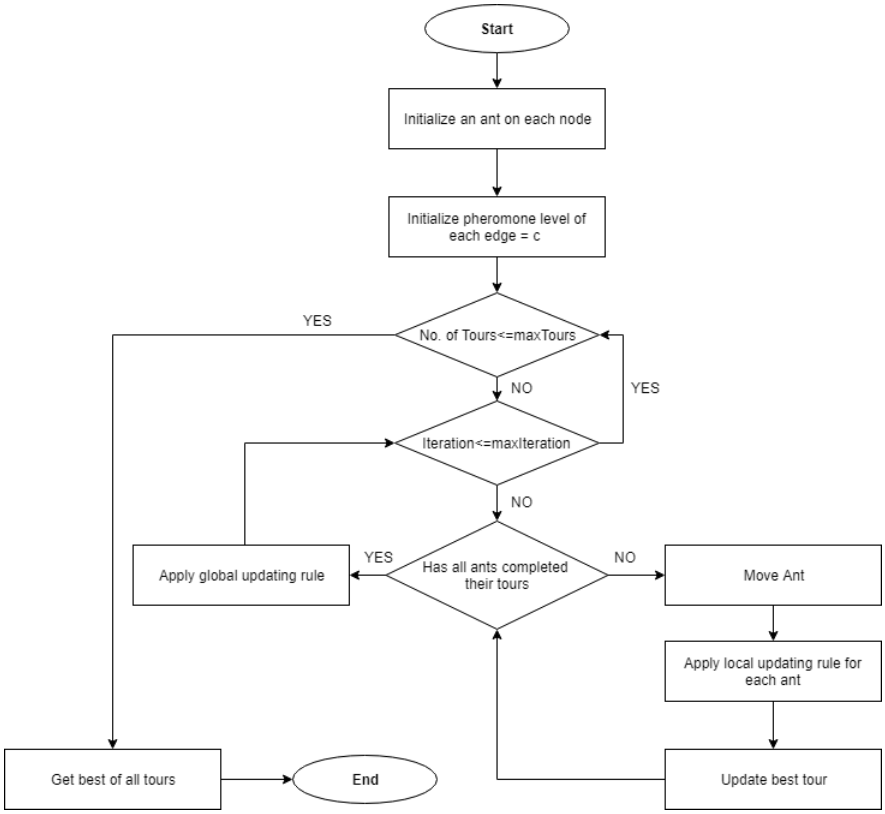

Figure 3.2: Flowchart that illustrated the use of ACS to solve the Gene Fragment Assembly Problem

3.4.1 Introduction to Local Search

If you have stuck through, you definitely have patience. Now a question may arise in your mind "Why Stage II?" .If we peek into section **??**, although it is early, we will notice that ACS plays a brilliant part in obtaining the gene fragment sequence in the stage I. However even after furthering our iterations, we reach an impasse in the sequences we obtain in terms of the overlap count. So we needed something more charismatic to improve the overlap count of the sequences. This is the very reason we chose to tread with local search.

Local search isn't a problem solving technique but an optimization technique. Local search, in most cases, will not completely solve the problem leaving you craving for further optimization. Local search algorithms operate using a single current node (rather than multiple paths) and generally move only to neighbors of that node. Although local search algorithms are not systematic, they have two key advantages: (1) they use very little memory—usually a constant amount; and (2) they can often find reasonable solutions in large or infinite (continuous) state spaces for which systematic algorithms are unsuitable [14].

There are many variants to the local search algorithm such as Hill Climbing search,

Stochastic Hill Climbing, Simulated Annealing and many more. However before we delve into deeper discussion, it would be a good idea to acquire an idea about the state-space landscape in figure 3.2. In figure 3.3, there are two things to consider "loca-

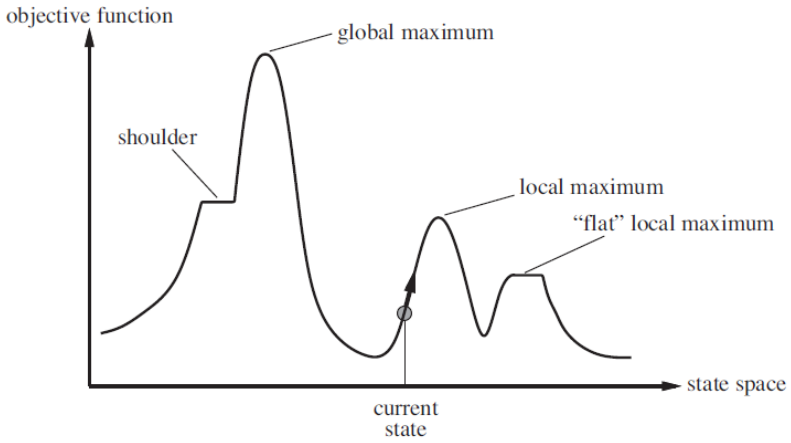

Figure 3.3: A one-dimensional state-space landscape where the aim is to find global maximum.Image from [16]

tion" and "elevation". Location defines the current state in the state-space landscape. Elevation can be further termed in to "cost" and "objective function". If it is cost, then we target to find lowest valley- global minimum. If it objective function, we then aim to find the highest peak- global maximum. A complete search algorithm will find the goal if it exists. However there are untoward circumstances related to this technique. The foremost being getting stuck in a local minimum or maximum depending the goal of the solution. Though there are ways to overcome such inconvenient situation once we encounter them.

3.4.2 Using LS to find longest sequence

The underlying local search approach used in Stage II was pretty straight forward. The entire quest for finding a better sequence has been laid out in figure 3.4. We have used randomized swapping of edges to maximize our objective function in (3.1).

Let us now delve deeper into this simple yet powerful approach to get proper insight about Stage II. Initially, we fed the algorithm with the sequences generated in stage I. Then each sequence is processed individually. Since these sequences were just paths generated by ants in Stage I. We could treat the two subsequent fragments as edges of a graph and the overlap count as the edge cost. For each sequence, we would select any two edges randomly and then swap their endpoints. If we had two edges, $u \to w$ and $v \to x$ and now we swap the endpoints to receive $u \to v$ and $w \to x$. We now compare the overlap count of the old sequence with the new sequence. If we receive better results we replace the old sequence with the new one else we restore the older

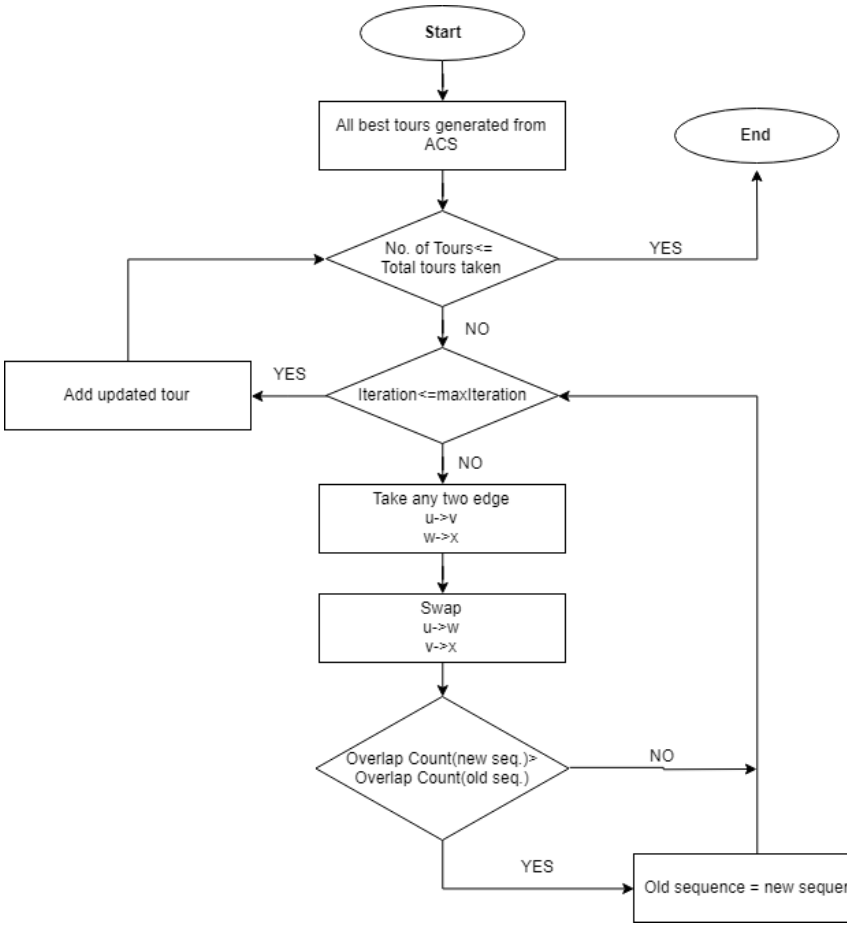

Figure 3.4: Flowchart of Local Search algorithm used to further develop the results from stage I.

sequence. This process was run for a fixed number of iterations to obtain better results than stage I.

3.5 Working Model

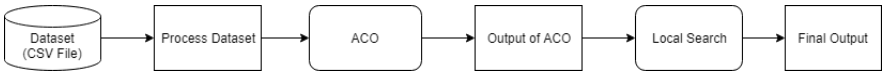

Figure 3.5: Working Model of the two stage hybrid algorithm

In this section we will briefly describe the working principle used by our proposed algorithm.

In figure 3.5, we see the illustration of the model used by our algorithm in this paper. Initially, the dataset is fed into the model in CSV format. Next, the system processes the received input. The processed input is then sent into the first stage of our model for finding the longest overlapping sequence as discussed in section 3.3.2. Once the first stage finishes its computation, we receive an array of DNA sequences. The received array of stage I is passed onto stage II for further processing. In stage II, we follow the process laid out in section 3.4.2 to varnish our results even more. After this session, we finally receive the single sequence with the best overlap count.

Chapter 4

Experimental Analysis

In this section we explain about the experimental results of our proposed algorithm which named is Solving Fragment Assembly Problem using hybrid 2-Stage Algorithm. It will evaluate the performance of the proposed algorithm. The results we found will be decorated by further investigating using statistical tests to clarify our observations. This will be designed as follows: Section 4.1 will describe the datasets and benchmark instances of different DNA fragments. In Section 4.2 we will discuss about the proposed algorithms approaches and the parameter we used to find out the results. In Section 4.3 we will compare our numerical results with other assemblers and on the last section **??** which will provide the summary of this analysis.

The implementation is performed using intel core i3 computer operating of 2.40GHz with 4GB of RAM, running windows 10 operating system.

4.1 Datasets

To check the efficiency and the accuracy of our algorithm we tested some benchmark instances. We examined 10 items of benchmark DNA fragments where we got overlaps between two fragments based on some parameters. After that we got a matrix of overlapping value of fragments. This benchmark instance of DNA fragments are obtained from National Center for Biotechnology Information (NCBI) web site. These sequences are as follow:

X60189 : A cluster of fibronectin type III and is 3835 base pairs long.

J02459 : Complete nucleotide sequence of Escherichia virus Lambda and is 20k base pairs long.

M15421 : A Homo sapiens apolipoprotein B gene, which is 10089 base pairs long.

BX842596 : A target sequence with this accession number we used BX842596(GI 38524253)in our work. Where this is the sequence of a Neurospora crassa DNA , which is 77292 bases long.

We obtained our overlap sequences using this benchmarks [15].Summarized version of those benchmarks are discussed in Table 4.1.

Table 4.1: details information of benchmark datasets [16].

Benchmark	Mean fragment length	Number of fragments	Original Sequence Length
x60189_4	395	39	3,835
x60189_5	286	48	3,835
x60189_6	343	66	3,835
x60189_7	387	68	3,835
bx842596_4	708	442	77,292
bx842596_7	703	773	77,292
m15421_5	398	127	77,292
m15421_6	350	173	10,089
m15421_7	383	177	10,089
j02459_7	405	352	20,000

4.2 Experimental Results

In this section we will discuss about our implemented algorithm experimental analysis, discussion and comparison between different algorithms which is being used to solve fragment assembly problems.

4.2.1 Parameter Tuning

One of the most important and major steps for any algorithm development is parameter tuning. The utmost optimal or maximize results of any algorithm is heavily affected by the values of different parameters which is used on algorithms. In our proposed algorithms we have initialize some parameters to find out the better results. To determine the right values of any parameters can be chooses manually observing the results acquired from different configuration. We have constructed the parameters values using manual observation.Table 2.1 show the parameters we have used so far for our proposed algorithms.

4.3 Performance Analysis

Here we will represent the results of our proposed algorithms and discuss the approaches we have applied on our proposed algorithm. First of all, to implement the algorithm here come a question how much ants we will placed to find the maximum overlaps on given datasets. Here on figure 4.1 show the results of manually configured ant size to find out the right values of ant size.

On this figure 4.1 we compare the number of ants against the fragment summation to find out the right value of how much ants we have applied on the ant colony optimization. To find out the maximum fragment overlaps we have manually configured

Figure 4.1: determining the number of ants

different types of parameters like shown on the figure4.1. It was a big challenge for us to provide good parameter values for the maximization.

Now we will discuss about the results that we have obtained for different 10 datasets which is being collected from NCBI website. To solve the FAP problems various algorithms are established to find out the overlaps between the fragments. There are some famous algorithms like PALS, POEMS, LKH etc which is very famous and effective for solving fragment assembly problem. Lets have a look on PALS and POEMS algorithms results and our proposed algorithms results on table-3 based on the Mallén-Fullerton and Fernandez-Anaya (2013)[16]. Refer to Mallén-Fullerton and Fernandez-Anaya (2013) for the details description of the algorithms.

On Table 4.2 we have tried to show the comparison of our 2-stage algorithm results with other two algorithms which is PALS and POEMS along with normal ant colony optimization and then hybrid-2 stage algorithm which have been optimized by local search operation.

Table 4.2: Results of 10 benchmark datasets along with proposed algorithm results

Benchmark	PALS	POEMS	ACO	Hybrid-2 stage
x60189_4	11,748	11,478	11,299	11,354
x60189_5	14,021		13,404	13,969
x60189_6	18,301		17,284	17,450
x60189_7	21,210	21,261	20,002	20,443
bx842596_4	225,782	227,233	208,084	210,297
bx842596_7	438,215	444,634	402,710	402,912
m15421_5	38,526	38,610	36,202	36,644
m15421_6	48,048		44,893	45,178
m15421_7	55,067	55,092	50,407	51,462
j02459_7	115,320	116,542	105,929	107,424

Using four sample of datasets to graphically view the comparison between three algorithms is on figure 4.2

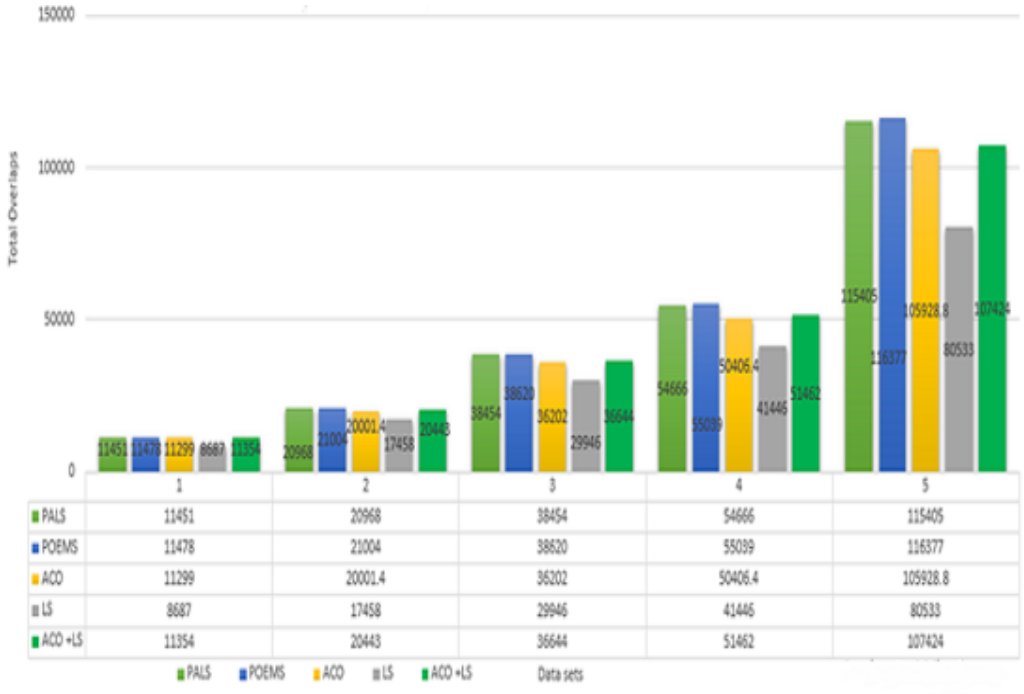

Figure 4.2: graphical view of three different algorithms results comparison

Here we can easily see the difference of results between different algorithms with the help of this graphical view.

Chapter 5

Summary, Conclusions, and Future Work

5.1 Summary

Through this work we have tried to solve the FAP using our 2-stage hybrid algorithm. We have tried our implementation against various benchmark instances to find the maximum overlap count for each dataset. Parameter tuning and changes made through graphical observations have allowed us to better our obtained results. We have compared our results against state of the art algorithms to get a better insight into our implementation.

5.2 Conclusions

Before the work is concluded, we would like to state that trying to solve the DNA FAP has struck a major realization that complex problems do not need complex solutions. ACO applied in stage-I of the proposed algorithm brought us very close to the benchmark results. Even though ACO is a nature inspired algorithm, it wasn't truly providing the randomness required to inch closer to the benchmark result. So we had to add Local Search in stage-II of the proposed algorithm. In stage-II, the difference between the fragment overlap count of the original sequence and our obtained sequence had lessened even more. It would be naïve to blame our failure on the algorithms rather it would be befitting to mention that we were unable to harness the true potential of the algorithms used. The proposed algorithm worked well in case of smaller datasets but in case of larger datasets the anomaly was very much evident. So, to conclude we haven't done justice to the problem yet and it needs a sincere effort to be solved through something more novel but simpler.

5.3 Future Work

We believe Genetic Algorithm [14] with its evolutionary approach would be able to further improve the obtained results.

Bibliography

[1] C. Li, "Dna fragment assembly algorithms: Toward a solution for long repeats," *Master's Projects. 98, San Jose State University.*, 2008. ix, 1, 2, 4

[2] M. D. et L.M. Gambardella, "Ant colony system : A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation, volume 1, numéro 1, pages 53-66,*, 1997. ix, 19, 20, 21

[3] H. Chial, "Dna sequencing technologies key to the human genome project," *Nature Education*, 2008. 1

[4] L. Li and S. Khuri, "A comparison of dna fragment assembly algorithms." pp. 329–335, 01 2004. 1

[5] M. Tammi, "The principles of shotgun sequencing and automated fragment assembly," *Center for Genomics and Bioinformatics, Karolinska Institutet, Stockholm, Sweden*, April 13 2003. 1, 19

[6] L. Li and S. Khuri, "A comparison of dna fragment assembly algorithms." *in METMBS, vol. 4, pp. 329-335*, 2004. 1

[7] J. Kubalik, P. Buryan, and L. Wagner, "Solving the dna fragment assembly problem efficiently using iterative optimization with evolved hypermutations," *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10. 213-214. 10.1145/1830483.1830522*, 2010. 2, 3, 15

[8] S. G. Miller JR, Koren S, "Assembly algorithms for next-generation sequencing data," *Epub*, march 6 2010. 9

[9] M. S. W. Pavel A. Pevzner, Haixu Tang, "An eulerian path approach to dna fragment assembly," *Proc Natl Acad Sci U S A. ; 98(17): 9748–9753. doi: 10.1073/pnas.171285098.* 11

[10] C.-S. Y. C.-W. T. Ko-Wei Huang, Jui-Le Chen, "A memetic particle swarm optimization algorithm for solving the dna fragment assembly problem." 12

[11] G. Kikuchi, S Chakraborty, "Heuristically tuned ga to solve genome fragment assembly problem," *2006 IEEE Congress on Evolutionary Computation, CEC 2006. 1491 - 1498. 10.1109/CEC.2006.1688485*, 2006. 13

[12] L. G. Alba E., "A new local search algorithm for the dna fragment assembly problem," *Lecture Notes in Computer Science, vol 4446. Berlin, Heidelberg*, 2007. 14

[13] B. Lin, Kernighan, "An effective heuristic algorithm for tsp," *Operations Research 21, 498–516*, 1973. 14

[14] S. J. Russell and P. Norvig, "Artificial intelligence,a modern approach third edition," 1995. 22, 31

[15] M. E. Y. Mohcin Allaoui, Belaid Ahiod, "A hybrid crow search algorithm for solving the dna fragment assembly problem, expert systems with applications," *doi: 10.1016/j.eswa.2018.02.018*, 2018. 27

[16] G. F.-A. G. M. Mallen-Fullerton, "Dna fragment assembly using optimization," *2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, pp. 1570–1577.*, 2013. 27, 28