Bee Algorithms for Solving DNA Fragment Assembly Problem with Noisy and Noiseless data

Jesun Sahariar Firoz ∗
A/EDA group
Department of CSE, BUET
Dhaka, Bangladesh
jesunsahariar@gmail.com

M. Sohel Rahman
A/EDA group
Department of CSE, BUET
Dhaka, Bangladesh
msrahman@cse.buet.ac.bd

Tanay Kumar Saha
A/EDA group
Department of CSE, BUET
Dhaka, Bangladesh
tanay@cse.jnu.ac.bd

ABSTRACT

DNA fragment assembly problem is one of the crucial challenges faced by computational biologists where, given a set of DNA fragments, we have to construct a complete DNA sequence from them. As it is an NP-hard problem, accurate DNA sequence is hard to find. Moreover, due to experimental limitations, the fragments considered for assembly are exposed to additional errors while reading the fragments. In such scenarios, meta-heuristic based algorithms can come in handy. We analyze the performance of two swarm intelligence based algorithms namely Artificial Bee Colony (ABC) algorithm and Queen Bee Evolution Based on Genetic Algorithm (QEGA) to solve the fragment assembly problem and report quite promising results. Our main focus is to design meta-heuristic based techniques to efficiently handle DNA fragment assembly problem for noisy and noiseless data.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—Sequencing

General Terms

Algorithm

Keywords

Bioinformatics, Genetic algorithms, DNA computing, Metaheuristics, Combinatorial optimization

1. INTRODUCTION

In DNA (Deoxyribonucleic Acid) fragment assembly problem (FAP), we are given a set of large number of DNA fragments with errors and are asked to find the correct sequence of the DNA by finding the permutation of fragments that best represents the DNA sequence. This problem finds its motivation from the limitation of current technology, which enables us to read only several hundreds of base pairs (bps) of a single DNA at a time. Consequently, we need to read fragments of the DNA but not the whole sequence at a time. During this process base pairs may be removed, misread or inserted.

To read the DNA fragments, a process named shotgun sequencing [31] is used. In this method, multiple copies of the DNA sequence are first generated through a process called amplification. Then we cut these sequences at random points keeping in mind that we can only directly read a sequence of several hundreds bps long. With these fragments, we try to reconstruct the overlapping DNA sequence as accurately as possible. This is called shotgun sequencing. A lot of tools have been invented to automate DNA sequencing. Among these tools, PHRAP [12], TIGR assembler [32], STROLL [7], CAP3 [13], Celera assembler [24] and EULER [26] may be cited. All of these tools focus on coping with different problems faced during fragment assembly.

To the best of our knowledge, the first work on solving the fragment assembly problem with meta-heuristics can be found in [20]. In this paper, approaches based on Ant Colony System (ACS) are proposed to solve FAP. Later, in [18], the authors presented several methods - a canonical genetic algorithm, a CHC (Cross generational elitist selection, Heterogeneous recombination and Cataclysmic mutation) method, a scatter search algorithm and a simulated annealing method - to solve accurately some problem instances that are 77K base pairs long. Since the work of [18], the problem of DNA fragment assembly has been tackled with different other heuristics and meta-heuristics in the literature [6, 8, 23]. However, the quest for new more accurate and faster techniques still continues.

Classical assemblers use fitness functions favoring solutions with strong overlaps between adjacent fragments in the layouts. But we also need to obtain an order of the fragments that minimizes the number of contigs, with the goal of reaching one single contig, i.e., a complete DNA sequence composed of all the overlapping fragments. PALS [6] (Problem Aware Local Search) is a simple, fast and accurate heuristic solution that is recently proposed with the objective of achieving this criteria. In PALS, the number of contigs is used as a high-level criterion to judge the whole quality of the results since it is difficult to capture the dy-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO ’12, July 7-11, 2012, Philadelphia, Pennsylvania, USA.
Copyright 2012 ACM 978-1-4503-1177-9/12/07 ...$10.00.
nematics of the problem into other mathematical functions. However, the calculation of the number of contigs is quite time-consuming, and this fact definitely precludes any algorithm to use such calculation. A solution to this problem was introduced in [6] as the utilization of a method which should not need to know the exact number of contigs and thus be computationally light. The key contribution of PALS [6] is to indirectly estimate the number of contigs by measuring the actual number of contigs that are created or destroyed when tentative solutions are manipulated. The authors in [6] used a variation of a heuristic algorithm for TSP for the DNA field, which does not only use the overlaps among the fragments, but also takes into account (in an intelligent manner) the number of contigs that has been created or destroyed.

In [8], the authors proposed a new method combining a general purpose meta-heuristic with a local search method specifically designed for this problem, namely, PALS of [6]. They presented a new approach of cellular genetic algorithms (cGA) that regulates the intensity on the search while solving a problem, outperforming the compared cGAs with static populations. Another paper [17] proposed evolutionary-based iterative optimization method called Prototype Optimization with Evolved Improvement Steps (POEMS) to solve the DNA fragment assembly problem. POEMS is an iterative algorithm that seeks for the best modification of the current solution in each iteration. The modifications are evolved by means of an evolutionary algorithm.

All of the above mentioned previous works consider operations on noiseless data. Recently, in [23], performance of various meta-heuristic based algorithms were discussed where a uniform random error model was introduced in the fitness score matrix. Very recently in [11] several hybrid meta-heuristics were proposed to tackle the fragment assembly problem with noisy data taking into consideration different error models (e.g. Sanger, 454 etc).

### 1.1 Our Contribution

In this paper, we focus on solving FAP for noisy and noiseless instances of DNA sequences. We have used fragments generated by GenFrag [9] to simulate noiseless instances. For noisy case, we first observe an important drawback of the recent work of [23]. The drawback of this technique is that no particular sequencing model was taken into consideration to incorporate the corresponding error model. To overcome this drawback and for the construction of a realistic read data set, here we use a sequencing simulator MetaSim [29]. In this simulator, the user is able to choose from different (adaptable) error models of current sequencing technologies (e.g. Sanger [21, 22], Roche’s 454 [19] and Illumina (former Solexa) [28]). In Sanger sequencing, the error model is based on the following two assumptions:

1. the probability of an error occurring at position $i$ of a read increases linearly with $i$, and
2. if an error occurs at position $i$, then with some fixed probabilities, it is either a substitution, a deletion or an insertion.

In 454 sequencing, the intensity of emitted light is used to estimate the length of homopolymers, i.e., runs of identical nucleotides in a sequence. During sequencing, the four DNA composing nucleotides are periodically flowed over the inserts to be sequenced. Within each flow, the intensity of the signal emitted reflects the number of nucleotides incorporated and thus the length of the homopolymer under consideration. For chemical and technical reasons, this signal is subject to fluctuations that lead to sequencing errors.

After generating realistic dataset with errors, we have implemented Artificial Bee Colony (ABC) algorithm and Queen Bee Evolution Based on Genetic Algorithm (QEGA) to solve the DNA fragment assembly problem as accurately as possible and compared the relative fitness achieved in each case.

As has been mentioned before, in most of the related works it was assumed that, the fragments being read are correct. We have eliminated this assumption by incorporating error models in generating artificial fragments to be sequenced and exploit the probabilistic behavior of ABC or QEGA so that worse individuals in the population are taken into consideration. In this way, the chance of completely eliminating a probably misread fragment is minimized and we are bound to find more realistic solutions. Simultaneously, we also analyze the behavior of these two algorithms for noiseless cases. Also, using forward and backward read technique of MetaSim [29], we have been able to emulate a more realistic input sequence mimicking experimental output. These facts give us a promising insight about the feasibility of using meta-heuristics to solve DNA fragment assembly problem.

The rest of the paper is organized as follows: In Section 2, we give an overview of DNA fragment assembly problem and error models used in fragment generation. In Section 3, we present the details of Artificial Bee Colony (ABC) algorithm and Queen Bee Evaluation based on genetic algorithm (QEGA) and the adaptation of these two algorithms to solve DNA FAP. Section 4 presents the experimental results achieved by executing our algorithms on several standard instances of DNA that are widely used in the literature. We briefly conclude in Section 5.

### 2. BACKGROUND

In this section, we present some background and preliminaries. Most of the definitions and procedures presented in this section follow from [18].

The input of the DNA fragment assembly problem is a set of fragments that are randomly cut from a DNA sequence. The DNA is a double helix of two anti-parallel and complementary nucleotide sequences. One strand is read from 5’ to 3’ and the other from 3’ to 5’ . There are four kinds of nucleotides in any DNA sequence, namely, Adenine (A), Thymine (T), Guanine (G), and Cytosine (C).

To further understand the problem, we need to know the following basic terminology:

- **Fragment**: A short sequence of DNA with length up to 1000 bps.

- **Shotgun data**: A set of fragments.
Prefix: A substring comprising the first \( n \) characters of a fragment \( f \).

Suffix: A substring comprising the last \( n \) characters of a fragment \( f \).

Overlap: Common sequence between the suffix of one fragment and the prefix of another fragment.

Layout: An alignment of a collection of fragments based on the overlap order, i.e., the fragment order in which the fragments must be joined.

Contig: A layout consisting of contiguous overlapping fragments, i.e., a sequence in which the overlap between adjacent fragments is greater than a predefined threshold.

Consensus: A sequence or string derived from the layout by taking the majority vote for each column of the layout.

To measure the quality of a consensus, we can look at the distribution of the coverage. Coverage at a base position is defined as the number of fragments at that position. It is a measure of the redundancy of the fragment data. It denotes the number of fragments, on average, in which a given nucleotide in the target DNA is expected to appear and is computed as follows [30]:

\[
\text{Coverage} = \frac{\sum_{i=1}^{g} \text{length of the fragment } i}{\text{target sequence length}},
\]

where \( g \) denotes the no. of fragments.

2.1 DNA Sequencing Process

To determine the function of specific genes, scientists read the sequence of nucleotides comprising a DNA sequence in a process called DNA sequencing. The fragment assembly starts with breaking the given DNA sequence into small fragments. To do that, multiple exact copies of the original DNA sequence are made. Each copy is then cut into short fragments at random positions (duplicates and sonicate phases). Then this biological material is “converted” to computer data (sequence and call bases phases). These steps take place in the laboratory. After the fragment set is obtained, traditional assembly approach is followed in the following phases in the given order: overlap, layout, and then consensus. In overlap phase, the best or longest match between the suffix of one sequence and the prefix of another are found. Layout phase consists of finding the order of fragments based on the computed similarity score. Consensus phase consists of deriving the DNA sequence from the layout. The most common technique used in this phase is to apply the majority rule in building the consensus.

2.2 Sanger Sequencing Error Model

In Sanger Sequencing [1] the following parameters concerning different probability values, \( P \) must be set because of the two assumptions mentioned earlier:

1. \( P \) (error at \( i \)) for positions \( i = 1 \) and \( i = 1000 \)
2. \( P \) (deletion | error)
3. \( P \) (insertion | error)
4. \( P \) (substitution | error) = 1 - \( P \) (deletion | error) - \( P \) (insertion | error)

2.3 454 Sequencing Error Model

For 454 sequencing [1] error model, let \( r \) denote the length of a given homopolymer. The emitted light intensity can be modeled by a normal distribution \( N(\mu, \sigma) \), with mean \( \mu = r \) and standard deviation \( \sigma = k \times \sqrt{r} \), where \( k \) is a fixed proportionality factor (\( k \approx 0.15 \)). Although basic statistics imply that the standard deviation should grow with the square root of \( r \), for the light intensity emitted during 454-sequencing, it is reported to be \( \sigma = k \times r \). In 454 model, a negative flow is a flow of nucleotides in which the sequence to synthesize is not elongated. Light intensities of negative flows follow a lognormal distribution, with mean \( \mu = 0.23 \), and standard deviation \( \sigma = 0.15 \). A random variable \( X \) is said to be lognormally distributed, if the random variable \( \ln(X) \) is normally distributed. Let \( \mu \) and \( \sigma \) be the mean and standard deviation of \( X \) and let \( m \) and \( s \) be the mean and standard deviation of \( \ln(X) \). Then,

\[
m = \ln\left(\frac{\mu^2}{\sigma^2 + \mu^2}\right), \quad s = \sqrt{\ln((\frac{\sigma}{\mu})^2 + 1)}
\]

The base-calling intensities of negative flows can be based on this and the misinterpretation of a base can be modeled as homopolymers of length = 1. A simulator takes the order of the sequencing flows into account. The nucleotides are cyclicly flowed in the order T, A, C, G and thus only two specific negative flows in a specific order are allowed after any given base. Now comes the base-calling process. As the details of the base-calling algorithm are not known in detail, and are presumably subject to constant improvements, there may occur a loss of accuracy here. First, the intersections of the density functions of the normal distributions for different homopolymer lengths \( r_1 \) and \( r_2 \) are calculated and stored in an intersection matrix \( M \). Then, these values are used to decide which homopolymer length is called.

2.4 Exact Error Model

The Exact Error Model is suitable for sampling reads without modifying any bases, i.e. in contrast to the other provided error models, no substitution, insertion or deletion is applied to the read sequences. But orientations may be changed during reading.

3. BEE ALGORITHMS

The Bee Colony meta-heuristic algorithms are nature-inspired algorithms based on various biological and natural processes observed in honeybees. Some of these algorithms are inspired by the foraging behavior of honeybee. One of them, Artificial Bee Colony (ABC) Algorithm, is a meta-heuristic designed upon the concept of the intelligent behavior of honey bee swarm. Honey bees use techniques like waggle dance that can be used to develop new intelligent search algorithms. Another class of Bee Colony algorithms is inspired by the queen bee evolution process and has used it to enhance the optimization capability of genetic algorithms. The queen-bee evolution makes it possible for genetic algorithms to quickly reach the global optimum as well as decreasing the probability of premature convergence.

3.1 Artificial Bee Colony (ABC) Algorithm

In the ABC algorithm [16], the colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts. A bee which waits at the dance area for making de-
cision to choose a food source depending on waggle dance of employed bee, is called an onlooker and a bee going to the food source visited by itself previously is named an employed bee. A bee which performs random search for food is called a scout. In the ABC algorithm (Algorithm 1), first half of the colony consists of employed artificial bees and the second half constitutes the onlookers. For every food source around the hive, there is only one employed bee. The employed bee whose food source is exhausted by the employed and onlooker bees becomes a scout.

For our DNA FAP, given a set of fragments, our target is to generate a permutation of fragments that minimizes the number of contigs and maximizes the fitness. The solution achieving these two objectives best represents the DNA sequence instance.

In our ABC algorithm, each of the food sources represents a permutation of fragments of a DNA sequence. The position of a food source represents a possible solution and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. Initially we generate the food source positions (i.e. permutations) randomly. We do not assume that the initial seeds are taken from the best of previous execution of any other algorithm. This is a more realistic assumption compared to [23] where the authors use seeding strategies to find good solution beforehand to exploit promising regions, for both noisy and noiseless data. Our algorithm is robust in the sense that, although we do not make any such preprocessing, our algorithm converges to almost same fitness compared to them in reasonable time.

After the initialization, the food sources are repeatedly searched by employed bees, onlooker bees and scout bees. A employed or onlooker bee produces a modification of the solution for finding a new food source and tests the nectar amount (fitness value) of the new solution. For the modification, we have used problem aware local search (PALS) [6] on the permutation µ under consideration. We have done so because, besides considering fitness value, we also take no. of contigs of a solution into consideration. In all cases, we have used a fitness function (calculation of the amount of nectar) that sums the overlap score for adjacent fragments \( f[i] \) and \( f[i+1] \) in a given solution. Let us denote the overlap score by \( w(f[i], f[i+1]) \) and fitness function by \( F_\mu \). Then,

\[
F_\mu = \sum_{i=0}^{g-2} w(f[i], f[i+1]). \tag{3}
\]

where \( g \) is the no. of fragments in the solution.

If the nectar amount of the new modified source is higher than that of the previous one, the bee memorizes the new solution and forgets the old one. Otherwise it keeps the position of the previous one. Onlooker bees choose the food source for modification probabilistically, depending on the nectar amount of the food source. The probability, \( p_\mu \) is calculated as follows:

\[
p_\mu = \frac{F_\mu}{\text{global}_{\text{max}}}, \tag{4}
\]

where \( \text{global}_{\text{max}} \) denotes the maximum fitness value among all food sources. If the nectar amount of a food source increases, the probability with which that food source is chosen by an onlooker increases, too. The dance of employed bees carrying higher nectar has higher probability of being selected by onlookers for exploitation.

**Algorithm 1 ABC Algorithm**

1: Initialize potential food sources for employed bees.
2: while Requirements are not met do
3:   Each employed bee goes to a food source in her memory and determines a neighbour source, then evaluates its nectar amount and dances in the hive
4:   Each onlooker watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source. After choosing a neighbour around that, she evaluates its nectar amount.
5:   Abandoned food sources are determined and are replaced with the new food sources discovered by scouts.
6:   The best food source found so far is registered
7: end while

### 3.2 Queen-bee Evaluation Based On Genetic Algorithm (QEGA)

Conventional genetic algorithm sometimes become unsuccessful to find a globally optimal solution within a limited no. of evolutions. To overcome this disadvantage, a queen-bee evolution based on genetic algorithm (QEGA) ([15], [27]) is employed for solving the DNA FAP.

**Algorithm 2 QEGA Algorithm**

Input: time \( t \), population size \( k \), populations \( P \), normal mutation rate \( \sigma \), normal mutation probability \( p_\mu \), strong mutation probability \( p_\sigma \), a queen-bee \( I_q \), selected bees \( I_m \).

Output: best fitness solution

1: Initialize \( P(t) \)
2: Evaluate \( P(t) \)
3: while Condition not met do
4:   \( t \leftarrow t + 1 \)
5:   Select \( P(t) \) from \( P(t - 1) \). So, \( P(t) = \{I_q(t - 1), I_m(t - 1)\} \).
6:   Recombine \( P(t) \)
7:   Crossover
8:   for \( i = 0 \) to \( k \) do
9:     if \( i \leq (\sigma \times k) \) then
10:        Mutate with \( p_\mu \)
11:      else
12:        Mutate with \( p_\sigma' \)
13:      end if
14:   end for
15:   Evaluate \( P(t) \)
16: end while

The population of QEGA consists of several permutations of fragments of a DNA sequence. To determine the fitness value of each individual in the population, we have used Equation 3 as the fitness calculating function in Steps 2 and 15 of QEGA (Algorithm 2).
There are two major differences between conventional genetic algorithm (CGA) and QEGA. Firstly, the parents $P(t)$ in CGA are composed of the $k$ individuals selected by a selection algorithm such as tournament selection. On the other hand, parents $P(t)$ in QEGA consist of the $\frac{k}{2}$ copies of a queen-bee $I_q(t - 1)$, where $q = \arg \max \{ F_m, 1 \leq m \leq k \}$ and $\frac{k}{2}$ copies of bees $I_m(t - 1)$ selected by a selection algorithm, where $1 \leq m \leq \frac{k}{2}$. Secondly, all individuals in conventional genetic algorithm are mutated with small mutation probability $p_m$, while in QEGA only a part of the individuals are mutated with normal mutation probability $p_m$ and the others are mutated with strong mutation probability $p'_m$. The ratio between $p_m$ and $p'_m$ is denoted by $\sigma$ in QEGA. Generally, $p_m$ is less than 0.1 and $p'_m$ is greater than $p_m$.

So, in QEGA, the fittest individual in a generation, cross-breeds with the bees selected as parents by means of a selection algorithm. This feature of queen-bee evolution reinforces the exploitation of genetic algorithms. That is, fitness of the offsprings mainly rely on the crossover operation and the fittest individual. Consequently, it also increases the probability of premature convergence. Nonetheless, the second feature of QEGA helps genetic algorithms search new space, i.e., it increases the exploration of genetic algorithms through strong mutation. These two features enable genetic algorithms to evolve quickly and simultaneously maintain good solutions. In other words, the queen-bee evolution makes it possible for genetic algorithms to quickly approach the global optimum as well as decreasing the probability of premature convergence.

Ordered two-point crossover has been used as crossover operator in Step 7 of Algorithm 2. In this scheme, given two parents, two random crossover points are selected partitioning them into a left, middle and right portion. Then ordered crossover is carried out in the following way: Child 1 inherits its left and right sections from Parent 1, and its middle section is determined by the fragments in the middle section of Parent 1 in the order these appear in Parent 2. A similar process is applied to determine Child 2. Notably, the Ordered crossover operator is a pure recombination operator [33]. For strong mutation, we have used problem aware local search (PALS) [6]. For normal mutation, we have done random mutation on the offsprings.

4. EXPERIMENTS

In this section, we present our experimental setup and the results obtained by our algorithms. The experiments were conducted in Ubuntu 11.04 running on an Intel core i3 Processor with 2GB RAM.

We have used GenFrag and MetaSim for generating noiseless and noisy artificial fragments respectively. For this, we collected the DNA sequences from NCBI [2, 3, 4, 5]. We give a summary on the different features of the datasets in Table 1.

4.1 Noisy Input Data Generation

We have used MetaSim [29] for noisy input data generation. The motivation behind using Metasim is briefly discussed as follows:

We have consulted different DNA sequence simulators like FASIM [14], GenFrag [9] and Celsim [25]. In [14], the authors have shown various problems associated with Celsim and GenFrag version 1.0 and 2.1. Mainly, these programs are designed on the assumption that fragments are equally distributed on genome. That is, clone sampling is uniformly carried out during fragment generation. Thus, these programs do not sufficiently reflect actual conditions of WGSS (Whole Genome Shotgun Sequencing). For FASIM, current release of the software is only available upon request. On the other hand, MetaSim is a freely available sequencing simulator for genomic and metagenomics. It can be utilized to simulate fragments of real read experiments by incorporating errors which occur at the layout phase, i.e., unknown orientation, base call errors, incomplete coverage, repeated regions, chimeras and contamination. Here we have used three error models, namely, 454, Sanger and exact error models considering configuration of all parameters. These configurations are presented in Table 2.

4.2 Score Matrix Calculation

As the exact orientation of the generated fragments are not predictable, to tackle the problem of unknown orientation, we have checked fragment overlapping in both forward and backward orientation during calculation of the score matrix. For example, let us assume that we want to calculate overlap between Fragment 1 and Fragment 2. We first calculate normal overlap between these two segments (suffix-prefix) and then compare the obtained value with the overlap value calculated from the reverse direction of Fragment 2 (suffix-suffix). We take the best value from the above two options.

4.3 ABC Control Parameters

ABC algorithm has a few control parameters. We set

---

### Table 1: Information of datasets. Accession numbers are used as the name of the instances

<table>
<thead>
<tr>
<th>Instances (abbreviated form)</th>
<th>Coverage</th>
<th>Mean fragment length</th>
<th>Number of fragments</th>
<th>Original sequence length (in bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>acin1 (a_1)</td>
<td>26</td>
<td>182</td>
<td>307</td>
<td>2170</td>
</tr>
<tr>
<td>acin2 (a_2)</td>
<td>9</td>
<td>1002</td>
<td>451</td>
<td>147200</td>
</tr>
<tr>
<td>acin3 (a_3)</td>
<td>3</td>
<td>1001</td>
<td>601</td>
<td>200741</td>
</tr>
<tr>
<td>acin5 (a_5)</td>
<td>2</td>
<td>1003</td>
<td>751</td>
<td>329968</td>
</tr>
<tr>
<td>acinv (a_7)</td>
<td>2</td>
<td>1003</td>
<td>901</td>
<td>426840</td>
</tr>
<tr>
<td>acin9 (a_9)</td>
<td>7</td>
<td>1003</td>
<td>1049</td>
<td>156305</td>
</tr>
<tr>
<td>x60189_4 (x_1)</td>
<td>4</td>
<td>395</td>
<td>39</td>
<td>3835</td>
</tr>
<tr>
<td>x60189_5 (x_2)</td>
<td>5</td>
<td>386</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>x60189_6 (x_3)</td>
<td>3</td>
<td>343</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>x60189_7 (x_4)</td>
<td>6</td>
<td>387</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>m15421_5 (m_1)</td>
<td>5</td>
<td>398</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>m15421_6 (m_2)</td>
<td>6</td>
<td>350</td>
<td>173</td>
<td>10089</td>
</tr>
<tr>
<td>m15421_7 (m_3)</td>
<td>7</td>
<td>383</td>
<td>177</td>
<td></td>
</tr>
<tr>
<td>j02459_7 (j_1)</td>
<td>7</td>
<td>700</td>
<td>352</td>
<td>48502</td>
</tr>
<tr>
<td>BX842596_1</td>
<td>4</td>
<td>708</td>
<td>442</td>
<td>77292</td>
</tr>
<tr>
<td>(b_1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BX842596_7</td>
<td>5</td>
<td>703</td>
<td>773</td>
<td></td>
</tr>
<tr>
<td>(b_2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Maximum number of cycles (MCN), i.e., maximum number of generation to 4000 and the colony size, i.e., population size to 256. The percentage of onlooker bees was set to 50% of the colony, the employed bees were 50% of the colony and the number of scout bees was selected as one. Notably, an increase in the number of scouts encourages the exploration whereas in onlookers of a food source increases the exploitation.

### 4.4 Results

Now, we present the results obtained by our ABC algorithm and QEGA for solving different DNA instances. We measure the quality of the solution by considering the fitness value achieved as well as the number of contigs. The significance of contig calculation is to ensure that the solution best represents a continuous assembled sequence. We have tested our algorithms for both noisy and noiseless fragments.

For noiseless instances, we set a cutoff value, i.e., required overlap between two adjacent fragments, to thirty. Table 3 shows the results obtained for noiseless instances by our algorithms as well as by simulated annealing (SA), Problem aware local search (PALS) and Genetic algorithm (GA). In Table 3, A, Q, S, P and G denotes ABC, QEGA, SA, PALS and GA respectively. In this case we have found that our implementation of ABC and QEGA perform very competitively with the best algorithm in the literature which is based on simulated annealing (SA) for noiseless instances [23]. Similar behavior is observed in comparison to PALS [6], a de facto standard to compare genome assemblers, in all cases. These experiments validate our algorithms’ good performance. Moreover, QEGA algorithm performs better than ABC algorithm for most of the instances. Notably, QEGA produces better results for longer instances. In terms of no. of contigs, QEGA does a better job at contig reduction than the ABC algorithm.

Next, We execute ABC and QEGA for noisy instances. It is to be noted that, in noisy instances, a lot of insertions, deletions or substitutions can occur. For instance, if we refer to Table 2, there are 5694 insertions, 1469 deletions and 1988 substitutions on j02452_7 instance in 454 error model. So, empirically we set the cutoff value for noisy instances to lower values but ensure overlapping adjacent fragments. The results obtained by our algorithms (ABC and QEGA) are presented in Table 4 along with three other algorithms from [11] namely Genetic Algorithms (GA), Genetic Algorithm with Simulated Annealing (GA+SA) and Genetic Algorithm with Hill Climbing (GA+HC). We have given the comparison of our result with [11] because the authors of [11]...
have used the same error models and parameters as ours to present results. More importantly, there is no way to compare our results with [23] because the authors of [23] have only told that they have introduced random error but there was no indication of how much error was being introduced. As opposed to them, we have clearly mentioned the parameters of our noisy dataset generation in Table 2. Instead of introducing random error in score matrix [23], we generated fragments with three error models (454, Sanger and Exact). We execute our algorithms on these instances. We also take into consideration that DNA FAP is an off-line problem. So, instead of emphasizing on execution time of the algorithms, we concern ourselves with assessment of the solution quality, based on number of contigs and fitness value.

As can be seen from Table 4, for most of the instances, QEGA outperforms all previously proposed algorithms as well as the ABC algorithm because of the elitist selection method imposed by QEGA. Our implementation of QEGA breeds fittest parent (i.e. queen bee) with other parents. The strong mutation in Step 12 of Algorithm 2 employs PALS to get the fittest mutants from the offspring. So we perform exploitation in the vicinity of the best solution of the current cycle. On the other hand, although ABC algorithm performs analogous action of mutation, it does not do so with the fittest ones. Exploitation can sometime leads to local minima and QEGA countermeasures this by random mutation. All the fitness values for QEGA and ABC algorithm shown in Table 4 are obtained for a single contig i.e. one continuous sequence with some exceptions. These exceptions can not reach single contig value and the final contig value for each of these instances are shown in brackets just beside their fitness in Table 4. For GA, GA+HC and GA+SA, the authors in [11] did not mention the final contigs obtained by these algorithms. So we are unable to comment about these three algorithms in terms of contig value.

Figure 1 and 2 show the progressive improvement of fitness values for the instance j02459_7 for noiseless and noisy data by applying QEGA. As can be seen from the figures, final fitness values for the noisy cases is significantly smaller than that of noiseless cases. Figure 1 also shows that QEGA is able to overcome any local minima as is evident from the transition of lower fitness value at almost 1100-th iteration to higher fitness value at almost 1200-th iteration.

5. CONCLUSION

In this paper, we have used Artificial Bee Colony (ABC) algorithm and Queen-bee Evaluation based on Genetic Algorithm (QEGA) to tackle the DNA fragment assembly problem for noisy and noiseless instances. In particular, we have recorded the results obtained by our algorithm, taking into consideration various standard error models normally used by DNA sequencing experiments. Previous works on DNA FAP report their results for noiseless cases mostly. We have observed that, although our algorithms perform very well for noiseless data just like others, for noisy instances their obtained fitness value decrease significantly for different error models. We have pointed out the fact that errors in DNA fragments significantly impact the performance of standard algorithms. Future works should take this into account. In future, we intend to design algorithms to achieve near-

<table>
<thead>
<tr>
<th>Instances</th>
<th>Error Model</th>
<th>Best Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>acin1</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>acin2</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>acin3</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>acin4</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>acin5</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>acin7</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>acin9</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>s60189_4</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>ml5421_2</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
<tr>
<td>j02459_7</td>
<td>454, Sanger</td>
<td>7735</td>
</tr>
</tbody>
</table>

Figure 1: Iteration vs Fitness graph for j02459_7 noiseless data using QEGA

optimal fitness for noisy instances. We also wish to implement the parallel versions of our algorithms.

6. REFERENCES

Figure 2: Iteration vs Fitness graph for j02459_7 noisy data(454 error model) using QEGA


