Bat Algorithm for Solving DNA Fragment Assembly Problem

Taher Muhammad Mahdee, Md. Habibur Rahman,Md. Mamunur Rashid

Abstract— DNA fragment assembly problem (FAP) is concerned with building the DNA sequence from several hundred (or even, thousands) of DNA fragments taken at random. Metaheuristic techniques are being used with very accurate results even for large problems. All existing methods rely on heuristics, since the DNA fragment assembly problem is NP-hard. Bat algorithms are used here to solve the DNA fragment assembly problem more efficiently.

Keywords— Bioinformatics, DNA fragment assembly, Metaheuristic, Bat algorithms, Genetic algorithms, Combinatorial optimization, Data mining

1 INTRODUCTION

DNA Fragment Assembly (DFA) is a system of trying to find the best order and orientation of a set of DNA fragments that can help finding the original DNA. With current technology like, gel electrophoresis, we are unable to accurately sequence DNA molecules that are more than 1000 bases long. In reality however, DNA contains much longer sequences. A human DNA for example, contains about 3.2 Billion nucleotides and we cannot read it all at once. To solve the problem, few techniques are developed. DNA molecules are first cut into smaller fragments at random locations and get sequenced directly. The overlapping fragments are then assembled back into the original DNA molecule. It is very important to perform this phase perfectly because later phases of any project with DNA depend on the accuracy of this part. DNA fragment assembly is a process that helps doing exactly this, assembling the fragmented DNA back to longer sequence. Most of sequence assembly algorithms are based on some variation of a greedy algorithm where the fragments are assembled by repeatedly merging the pair of fragments with the highest overlap (similarity score) according to a specific and complex criterion. Greedy methods obtain good results for small to medium sequences but fail to deal with large genome sequencing projects. Meta-heuristic techniques are being used with very accurate results even for large problems. All existing methods rely on heuristics, since the fragment assembly problem is NP-hard. The most popular techniques are evolutionary algorithms, ant colony systems, and simulated annealing.

- *Taher Muhammad Mahdee is currently teaching in Bangladesh Army University of Science and Technology as Lecturer. His email address is t.mahdee@gmail.com .*
- *Md. Habibur Rahman is pursuing his M.Sc. in CSE from Bangladesh University of Engineering and Technology. His email address is talha629@gmail.com*

Though these algorithms achieve good result in case of sequencing DNA fragments, however, expected accuracy is yet to receive. In this regard, for achieving more accuracy, we have applied a new meta-heuristic algorithm (BAT Algorithm) in DNA fragment assembly problem.

2 RELATED WORK

2.1 Genetic Algorithm

In 1995, Rebecca Parsons Stephanie Forrest and Christian Burks worked with various genetic algorithm operators that can help with the permutation problem associated with the Human Genome Project in order to assemble the DNA sequence fragments [1].

In [2], a GA using a permutation representation is studied to solve this problem with different recombination operators. Edge Recombination (ER) and Order crossover (OX) are used along with some permutation operators, such as Cycle crossover (CX) and Partial Mapped crossover (PMX). They also analyze different seeding strategies to generate the initial population which is an important issue in building successful GAs. To do so, they incorporate solutions generated by a 2- opt heuristic and a greedy approach to the initial population in order to improve the final accuracy and efficiency of the algorithm.

From the study and analysis of all these components of the genetic algorithm they observe that the 2-opt heuristic to generate the population is very beneficial since it allows improving the fitness quality without increasing the execution time. From the recombination operator point of

view, the edge recombination achieves the best results, although it results in a longer execution time for random and 2-opt seeding strategies. On the other hand, cycle and order crossover operators are the best options when the greedy strategy is used.

FAP can be efficiently solved using meta-heuristics with success but still has problems in presence of noise in the data while being input or while being searched and more so in large instances. Gabriela Minetti et al. [3] propose a parallel and hybrid meta-heuristics PH-PALS, combining Problem Aware Local Search (PALS) technique with Simulated Annealing (SA) for solving noisy instances of FAP. The hybrid approach they proposed resulted better performance in both cases of very large non-noisy as well as noisy cases compared to Simulated Annealing or PALS.

In [4], four heuristic algorithms are designed, implemented, tested and then compared. In genetic algorithms, three operators: selection, crossover, and mutation are applied to a population of individuals to create a new population. They use ranking selection mechanism, in which the GA first sorts the individuals based on their fitness values and then selects the individuals with the best fitness score until the specified population size is reached. Two crossover operators: order-based and edge-recombination are implemented to allow partial solutions to evolve in different individuals and then combine them to produce a better solution.

They use the swap mutation operator for modification of single individual, which randomly selects two positions from a permutation and then swaps them.

The first step of the algorithm helps building the Best Set of Maximum Weight Contigs (BSC) having the complexity of O(n2 l2). Here n is the Fragments count and l is the fragments length averaged. The next step of the algorithm is for sorting the Maximum Weight Contigs (MWC) of BSC considering the contig overlaps order with complexity O(m2 l2). In this case, m is the number of MWCs.

This process calculates the contig overlaps instead of fragment overlaps In order to get the both of these advantages. One is, considering just true overlaps and making sure of finding the orientation of the fragments.

The third heuristic they applied is Structured Pattern Matching Algorithm, which is based on a technique called hybridization Fingerprinting. Here, the task is divided into three phases. In first phase, they randomly select short probes from each fragment, and then use exact pattern matching in determining the relative positions of the input fragments. Thus, each fragment is represented as an ordered set of probes and associated inter-probe distances rather than a sequence of nucleotides. A detailed map is constructed in second phase to show how fragments are ordered and how they align. The third phase is used to determine the sequence. The time complexity of the Structured Pattern Matching algorithm is approximately linear in the length of the target sequence.

The fourth method they applied is Clustering Heuristic Algorithm with traditional three steps: overlap, layout, and consensus. They use the semi-global alignment algorithm to find all possible pairwise overlaps. After determining the overlaps, they use a greedy heuristic in the layout phase to find the multiple sequence alignment among a set of fragments. The pair of fragments with highest overlap are taken as the starting point. The layout is constructed by successively adding the fragment that has the highest overlap with the assembled fragments, until no fragment is left.

After analyzing the result, they conclude that for smaller data sets, all four algorithms got the same result in approximately the same running time. However, the results and the performance vary as the data sets become larger. For larger data sets, i.e., 50 or more fragments, the performance of the algorithms ranking from best to worst is: Structured Pattern Matching Algorithm, Clustering Heuristic Algorithm, Genetic Algorithm, and finally the Greedy Algorithm.

2.2. Swarm intelligence

The paper [5] used swarm intelligence to solve this problem which is based on the study of behavior of simple individuals (e.g. ant colonies, bird flocking, and honey bees, animal herding) that mimics the behavior of swarms of social insects or animals.

In [6], Stochastic Diffusion Search (SDS) algorithm is used. This algorithm also belongs to the category of swarm

intelligence and is based on mimicking the foraging behavior of one type of ants Leptothorax acervorum.

In 2003, Meksangsouy and Chaiyaratana proposed ant colony optimization. The goal of the search was to find the right order and orientation of each fragment to create a consensus sequence [7].

There are a few proposed solutions different than these in 2011 to solve the DNA sequence assembly problem using Particle Swarm Optimization (PSO) using Shortest Position Value (SPV) rule. [8]. In 2012 Firoz analyzed and discussed the performance of two swarm intelligence based algorithms namely Artificial Bee Colony (ABC), and Queen Bee Evolution Based on Genetic Algorithm (QEGA) to solve the fragment assembly problem [9]. In 2013 Fernandez-Anaya et al. designed a nature inspired algorithm (PPSO+DE) based on Particle Swarm Optimization and Differential Evolution [10].

3 PRELIMINARIES

In this section, we present some background and preliminaries. Most of the definitions and procedures presented in this section follow from [11].

The input of the DNA fragment assembly problem is a set of fragments that are randomly cut from a DNA sequence.

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

1. Fragment: A short sequence of DNA with length up to 1000 bps.
2. Shotgun data: A set of fragments.
3. Prefix: A substring comprising the first characters of a fragment.
4. Suffix: A substring comprising the last characters of a fragment.
5. Overlap: Common sequence between the suffix of one fragment and the prefix of another fragment.
6. Layout: An alignment of a collection of fragments based on the overlap order, i.e., the fragment order in which the fragments must to be joined.
7. 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.
8. 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 [12]:

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

4 PROPOSED ALGORITHM AND EXPERIMENTAL RESULT

DNA Fragment Assembly is an NP-hard problem. Many algorithms have already been used to find accurate DNA sequence from a collection of fragments. However, satisfactory result has not been achieved yet. Still now, it is one of the crucial challenges faced by computational biologists. In recent years it has been noticed that nature provides hints to solve many critical problems. Many difficult problems have been solved by using nature inspired algorithms. The vast majority of heuristic and meta-heuristic algorithms have been derived from the behavior of biological systems and/or physical systems in nature. For example, particle swarm optimization was developed based on the swarm behavior of birds and fish [13, 14], while simulated annealing was based on the annealing process of metals [15]. New algorithms are also emerging recently, including harmony search and the firefly algorithm [16]. The former was inspired by the improvising process of composing a piece of music [17], while the latter was formulated based on the flashing behavior of fireflies. Bat algorithm [18] is another nature inspired algorithm which was introduced in 2010. It provides a new meta-heuristic method, based on the echolocation behavior of bats. We use this algorithm to solve DNA Fragment Assembly problem. We analyze the performance of bat algorithm to solve the fragment assembly problem and compare it with existing results.

4.1 Bat Algorithm

1. All bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers in some magical way;
2. Bats fly with no defined sequence having velocity v_i at position x_i having a fixed frequency f_{min}, that changes in wavelength and amplitude A_0 to look for the target. They can change the wavelength (or frequency) at will in their produced pulses and tune the rate of pulse emission r 2[0, 1], depending on distance of their prey;
3. Even though, the sound amplitude can be modulated in varieties of ways, we assume that this amplitude changes from a high (positive) A_0 to a minimum constant value A_{min}.

Algorithm 1
Bat Algorithm

Pseudo code of the bat algorithm (BA).

Objective function f(x), x = (x1,……,xd)^T
Initialize the bat population xi (i = 1, 2,…..,n) and v_i
Define pulse frequency f_i at x_i
*Initialize pulse rates r_i and the loudness A*i
while** (t < Max number of iterations) **do
Generate new solutions by adjusting frequency, and updating velocities and locations/solutions
if** (rand > r_i) **then
Select a solution among the best solutions
Generate a local solution around the selected best solution
end if
Generate a new solution by flying randomly
if** (rand < A_i & $f(x_i) < f(x^*)$) **then
Accept the new solutions
*Increase r_i and reduce A*i
end if
Rank the bats and find the current best x
end while
Post process results and visualization

4.2 Proposed algorithms

We implement the algorithm inspired from Bat algorithm to solve the DNA fragment assembly problem with some changes which is similar to Genetic algorithm. To generate the local solution from the best solution we used crossover and mutation. To generate a new solution we use random order of thing of the fragments.

For fitness calculation we used a fitness function 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. So,

$$F = \sum_{i=0}^{n-2} W(f[i], f[i+1])$$

where n is the no. of fragments in the solution.

4.3. Experimental setup

In this section, we present the results obtained by executing Bat Algorithm for solving DNA Fragment Assembly Problem (FAP). We have used the DNA sequences available at NCBI [19, 21]. We give a summary on the different features of the datasets in Table I. To generate the fragments from the DNA sequences, we have used the DNAgen instances [22] tool. We use these datasets in the above mentioned algorithms implemented using c++. The environment was simulated in Mac OSX 10.11 running on an Intel core i5 Processor with 8GB RAM.

4.3.1 Algorithm Implementation

We implemented this algorithm from scratch using C++. For data representation we use STL library. We use all unique random number in a range to generate an individual. We have used ordered crossover and swap mutation in our algorithms. Swap mutation randomly picks two points and swap their positions. This operation run several time based on a random number generated each time the operation is called. Ordered crossover and swap mutation with probability for genetic algorithm.

4.3.2 Result

Table 2 presents the results obtained for solving the FAP. We have executed 10000 iterations of BA for each instance. We present the best fitness obtained in each case. At Table 3 in conjunction with Figure 1, we can see that the fitness value improves significantly with the increment of no. of

iterations. This suggests that, to get higher accuracy, we need to increase the number of iterations of if time permits.

Table 1
Information of Dataset

Instances	Coverage	Mean fragment length	Number of fragments	Original sequence length
acin1	26	182	307	2170
acin2	3	1002	451	147200
acin3	3	1001	601	200741
acin5	2	1003	751	329958
acin1	2	1003	901	426840
acin9	7	1003	1049	156305

Table 2
Best Fitness Obtained

Instances	Best Fitness Obtained
acin1	19698
acin2	90766
acin3	135920
acin5	8969
acin7	321819
acin9	249965

Table 3
Fitness Obtained for Acin1 By BA

No of Iterations	Fitness Obtained
10000	19698
20000	23624
30000	25578
40000	26740
50000	27361

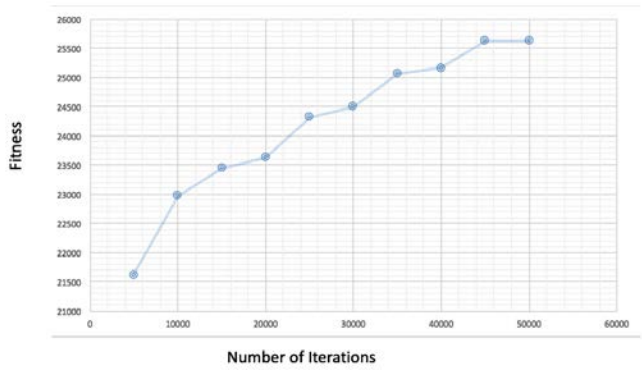

Fig 1: Relation between number of iterations and fitness for acin1 by BA.

5 CONCLUSION

We have solved the DNA Fragment Assembly Problem using Bat Algorithm. The implementation isn't finely tuned yet. Exploration and exploitation function needs more improvement. Tweaking operation can also be improved. In this work, only noiseless Data is used to solve the FAP problem. There are still opportunities to work a lot more with this solution.

Table 4
Overall results comparing BA algorithm with other algorithms.

Item Benchmark	LKH [14]	PPSO [13]	QEGA [2]	SA [2]	PALS [15]	SAX [15]	FF	BA
acin1	47618	47264	47115	46955	46876	46865	45160	19698
acin2	151553	147429	144133	144705	144634	144567	147460	90766

acin3	167877	163965	156138	156630	156776	155789	164652	135920
acin5	163906	161511	144541	146607	146591	145880	162915	8969
acin7	180966	180052	155322	157984	158004	157032	179913	321819
acin9	344107	335522	322768	324559	325930	314354	333815	249965

6 REFERENCES

[1] Rebecca J. Parsons et al., "Genetic algorithms, Operators, and DNA Fragment Assembly", October 1995, Volume 21, Issue 1, pp 11-33.

[2] Gabriela Minetti et al., "Seeding strategies and recombination operators for solving the DNA fragment assembly problem", April 2008.

[3] Gabriela Minetti et al., "An improved trajectory-based hybrid meta-heuristic applied to the noisy DNA Fragment Assembly Problem", February 2014.

[4] Lishan Li and Sami Khuri, "A Comparison of DNA Fragment Assembly Algorithms"

[5] Blum,C., Li,X., "Swarm intelligence in optimization", Springer(2008).

[6] Al-Rifaie, M.M., Bishop, M., "Stochastic diffusion search review", Journal of Behavioral Robotics, vol. 4(3), 2013.

[7] Meksangsouy, P., Chaiyaratana, N., "DNA fragment assembly using an ant colony system algorithm", Evolutionary Computation, 2003. CEC03. The 2003 Congress on. vol. 3, pp.17561763. IEEE (2003).

[8] Verma,R.S.,Singh,V.,Kumar,S., "DNA sequence assembly using particle swarm optimization", International Journal of Computer Applications 28 (2011).

[9] Firoz, J.S., Rahman, M.S., Saha, T.K., "Bee algorithms for solving DNA fragment assembly problem with noisy and noiseless data", Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference. pp. 201208. ACM (2012).

[10] Mallen-Fullerton, G.M., Fernandez-Anaya, G., "DNA fragment assembly using optimization. In: Evolutionary Computation (CEC)", 2013 IEEE Congress on. pp. 15701577. IEEE (2013).

[11] Yang, X.S., "Fire Fly algorithms for multimodal optimization", Proc. 5th Int. Conf. on Stochastic Algorithms: Foundations and Applications SAGA'09, Springer, Berlin, Heidelberg, 2009, 169178.

[12] J. Setubal and J. Meidanis, Fragment assembly of dna, Introduction to Computational Molecular Biology, pp. 105139, 1997.

[13] Kennedy, J. and Eberhart, R.: Particle swarm optimization, Proc. IEEE Int. Conf. Neural Networks. Perth, Australia, 1942-1945 (1995).

[14] Kennedy, J. and Eberhart, R., Swarm Intelligence. Academic Press, (2001).

[15] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P.: Optimization by simulated annealing. Science, 220, 671-680 (1983).

[16] Geem, Z.W., Kim, J. H., Loganathan, G. V.: A new heuristic optimization algorithm: Harmony search. Simulation, 76, 60-68 (2001).

[17] Yang, X.-S.: Nature-inspired Meta-heuristic Algorithms. Luniver Press, (2008).

[18] Yang, X. S., "Bat algorithm: literature review and applications." International Journal of Bio-Inspired Computation, 5(3), 141-149 2013.

[19] http://www.ncbi.nlm.nih.gov/ nuccore/178817?report=fasta(M15421.1), [Online; accessed 20-November-2011].

[20] http://www.ncbi.nlm.nih.gov/ nuccore/34645?report=fasta(X60189.1, [Online; accessed 20-November-2011].

[21] http://www.ncbi.nlm.nih.gov/nuccore/215104? report=fasta(J02459.1. [Online; accessed 20-November-2011].

[22] M. L. Engle and C. Burks, Artificially generated data sets for testing dna sequence assembly algorithms, Genomics, vol. 16, no. 1, pp. 2868,1993.