Annales Univ. Sci. Budapest., Sect. Comp. 42 (2014) 69–81

APPLYING THE FIREFLY APPROACH TO THE DNA FRAGMENTS ASSEMBLY PROBLEM

Anna Bou Ezzeddine (Bratislava, Slovakia)

Stefan Kasala (Bratislava, Slovakia)

Pavol Navrat (Bratislava, Slovakia)

Dedicated to András Benczúr on the occasion of his 70th birthday

Communicated by Attila Kiss

(Received June 1, 2014; accepted July 1, 2014)

Abstract. In Informatics we sometimes come across problems, which cannot be solved by common algorithms. Other times, the devised algorithms are too complex for real use. An alternative solution can often be found in the use of metaheuristics. One of the metaheuristics subgroups are algorithms inspired by nature. Their main inspiration is nature itself. They imitate processes from life. Social insects like ants or bees are good examples. At the first sight they are very simple organisms. When we look closer, however, their overall behavior is amazing, as is their organization in a swarm when working towards their common goal. Application of these algorithms can lead to interesting solutions in many different fields, especially when no other viable solution is available.

DNA assembly is one of such problems. The problem is to assemble DNA from fragments read by some DNA sequencing technology, since current technologies are not able to read the whole DNA sequence, only much shorter fragments. We propose a possible solution to the DNA assembly problem by use of a biologically inspired algorithm that imitates fireflies. We adapted algorithm for this problem and designed new algorithm operators. We implemented the proposed solution in a prototype. Finally, we successfully verified the algorithm on GenFrag and DNAgen benchmark instances of DNA problem.

Key words and phrases: DNA, assembly, firefly algorithm, nature inspired computing

2010 Mathematics Subject Classification: 68T20

The project was partially supported by the Slovak Research and Development Agency under the contract No. APVV-0208-10 and the Scientific Grant Agency of Slovak Republic, grant No. VG 1/0752/14.

1. Introduction

Analysis and data processing are often required when conducting research in various scientific disciplines. Often, informatics becomes involved. Interdisciplinary fields emerge, such as bioinformatics. One of the most recent problems in bioinformatics is the DNA assembly. Knowing a DNA structure is of fundamental importance in researching how to prevent many illnesses and how to cure them. Due to progress in technology, it is currently possible to read very big numbers of very small DNA fragments faster and faster, using e.g. "shotgun sequencing" [9]. The problem is that we do not know the original positions of fragments in the DNA sequence. To solve this computationally difficult problem, many algorithms were devised. Their main goal is to reconstruct the original structure of the DNA.

One of the possible approaches is to use biologically inspired algorithms (BIA). BIA are stochastic search and optimization techniques based on principles of collective behavior and self-organization. They are often inspired by social behavior of living organisms. They were successfully applied to many fields of computationally difficult problems, where solutions using conventional algorithms are not possible. DNA assembly is such a problem.

2. Problem definition

The problem of DNA assembly can be characterized by the input and the corresponding output. Input is a huge amount of small fragments, which are essentially strings composed of only four characters A, G, C and T. Output should be the original DNA sequence, which is a permutation of inputs. It is a combinatorial *NP-hard* class problem [11].

The current technology (sequencing of second generation, "shotgun sequencing"), allows to read many DNA fragments in parallel very fast, but the fragments must be short (of the order of hundreds). Thus the original DNA must first be fractured. If not for this, there would be no problem of assembly. Moreover, the way the fracturing is done has an unpleasant consequence that information on a fragment's position in the original sequence is lost. To be able to reconstruct completely the whole DNA we need to reach proper coverage of DNA sequence by a sufficient number of fragments [11], where sufficiency is sought by redundancy. Redundancy is also needed to allow fragment overlaps.

One way of viewing the problem is finding the shortest common super-

string by constructing Euler super-path [10]. Regardless the approach taken, the problem is $O(NP)$. However, the real problem, not viewed so abstractly, is tied with additional complications. DNA structure consists of two complementary strings and the fragments come from either of them. In a genome of an organism, repetitions of variable length and frequency are common. Reading process is error prone. Its output can contain with some probability various errors that need to be detected and repaired. The main problems can be summarised as follows [5]. (1) unknown direction - overlaps between two fragments depend on their order, (2) reading errors - current technology is error prone and 1 to 10 % error rate is common, (3) insufficient coverage - fragments are read randomly, the genome is not guaranteed to be covered completely, (4) repetitions - repetitions that are bigger than readings cannot be detected, (5) chimeras and contamination.

3. Standard approaches

Vast majority of assemblers are based on 2 standard approaches [8].

Overlap-layout-consensus method was used in the Sanger project. In the first phase, *overlap*, graph of overlapping fragments is created (e.g. using suffix trees). After the first phase graph contains much duplicate information. In the *layout* phase the graph is simplified by many operations, e.g. removing edges that skip nodes, where the other path already exists. Also the contigs are separated. The *consensis* stage relies on the high coverage of DNA. It picks most likely nucleotide sequence for each contig. This phase can also repair errors caused by the reading technology.

The most modern assemblers are based on the second approach, *De Bruijn Graphs* [9]. Each node represents a unique string of length k, which can be found in some input sequence or its complement. An oriented edge links two nodes “aA” and “Ab” in the case that string “aAb” can be found in original sequence. DNA assembly is the shortest path of graph that consists of all the nodes. For double strand character of DNA the bi-directed graph is used. The tools based on this principle are for example *Euler*, *Velvet*, *ABySS*, *AllPaths*, and *SOAPdenovo*. The tools differ mainly in internal memory representation of data, dealing with a pair of readings and reading errors.

4. BIA approaches

Optimization in DNA assembly, increasing the quality of assembly output leads to the use of alternate methods such as BIA approaches [3]. One group of BIA are algorithms inspired by swarm intelligence.

In [2] authors combined 2 BIA swarm algorithms, *Artificial Bee Colony* (ABC) and *Queen Bee Evolution Based on Genetic Algorithm* (QUEGA). They successfully applied algorithms for assembly of errorless data and data with some rate of artificial errors. The food in ABC and individual in QUEGA was one solution represented by sequence permutation. With the set of DNA fragments in input, the optimization problem was to minimize number of contigs and maximize the overlap score of permutation. The algorithms do not need any data preprocessing. The authors used problem aware local search (PALS). They used GenFrag and MetaSim for data generation. They observed comparable results for errorless data with other BIA approaches. For data with some degree of errors, they observed better performance with QUEGA.

Another popular BIA optimization is based on ants behavior. It is often demonstrated on travelling salesman problem, which can be viewed also as an abstraction of the DNA assembly. Authors of [7] applied the ant colony optimization (ACO) to the DNA assembly problem. The cities are fragments and the city distances are analogy to the fragment similarity. The goal is to travel the shortest path over all the cities. In DNA assembly world it means to find the shortest string of all the fragments. The algorithm was tested on the subparts of human genome that was cut to fragments with no errors. It performed better if numerous contigs were composed.

5. Firefly algorithm

Our approach is based on the algorithm inspired by fireflies - *Firefly algorithm* (FA). The algorithm was first published in 2009 by its creator Xin-She Yang [12]. The algorithm is inspired by the fact that fireflies are able to produce light to become attractive. In real life males and females are attracted by lights for the purpose of reproduction. The main principles are [12]:

- the fireflies are bisexual;
- attractiveness is proportional to firefly light intensity, the fly with less

intensive light moves always to the fly with more intensive light;

- the light intensity decreases proportionally to distance between two flies, caused by light absorption;
- light intensity is characterized by an objective function. Objective function is similar to fitness function, which is metric for solution quality in genetic algorithms.

Pseudocode for general algorithm [12]:

1. initial parameters are defined: fireflies count, number of moves M, iteration limit N, light absorption $gamma$;
2. generate initial population P;
3. repeat N times or until desired solution achieved:
   (a) for every firefly F in population P:
      i. find the most attractive F_A;
      ii. if more attractive is not visible for defined $gamma$, move random M times;
      iii. otherwise move to the more attractive M times;
   (b) evaluate new flies based on their lightness, choose fireflies for the next iteration.

Each firefly is one problem solution, in our case a permutation of the DNA fragments. The distance between fireflies is a measure of difference between fireflies. Light absorption is constant, usually a value from 0.01 to 100. It describes how far a firefly can see in the solution space. If set too low, firefly can see all the other fireflies and algorithms are similar to particle swarm optimization. If set to high value, fireflies are blind and algorithms become random search.

The main idea of the algorithm is to improve solution using similar but better solution. Yang claims that it finds extremes in an effective way and also with the global optima between them. It performs better and converges faster compared to genetic algorithm and particle swarm optimization.

The attraction in distance r is defined as [12]:

$$\beta(r) = \beta_0 e^{-\gamma r^2}, \tag{5.1}$$

r - distance,
B_0 - lightness of firefly = fitness,

γ - light absorption constant, optional number (mostly between 0,01 and 100).

The algorithm is primarily designed for continuous optimization problems. Its performance was illustrated with *Michalewitz function* for 2 independent variables. It needs to be modified to meet combinatorial problem requirements. The possible solution was illustrated in [4]. The movement part of algorithm is an abstraction of solution modification. In our solution we designed components of the algorithm as follows.

5.1. Algorithm modifications

Firefly:
One firefly is a permutation of all available input DNA fragments. In the case of a continuous problem the firefly was defined by values of optimized problem parameters.

Movement:
Movement should express various length and direction. We designed two algorithms. The first one is pure random movement. We take random number N from uniform distribution depending on distance between 2 fireflies. Then the moving firefly is N times randomly changed.

The second one expresses movement characteristics in a better way. In addition to the length it is also directed. We called it *SequenceConstructedMovement* (SCM). It is inspired by the SCX operator used in genetic algorithm for combinatorial problems [1]. It tries to compose a new, better permutation by adding the subsequences of the firefly we are moving to. By using random factors with probabilities we can control the ratio of edges from moving firefly, toward firefly and new edges. The algorithm pseudocode can be found in the Picture 1.

Distance:
To determine distance between firefly A and firefly B we devised 2 algorithms. The first one is based on one to one comparison of permutations. Distance is percentage of object in different positions to all positions in permutation.

The second one uses a relative position of fragments in permutation, the edges between fragments. It looks at the successor of compared permutation objects. If they are not the same, the counter is incremented. The result is relative to the permutation length.

Attractiveness:

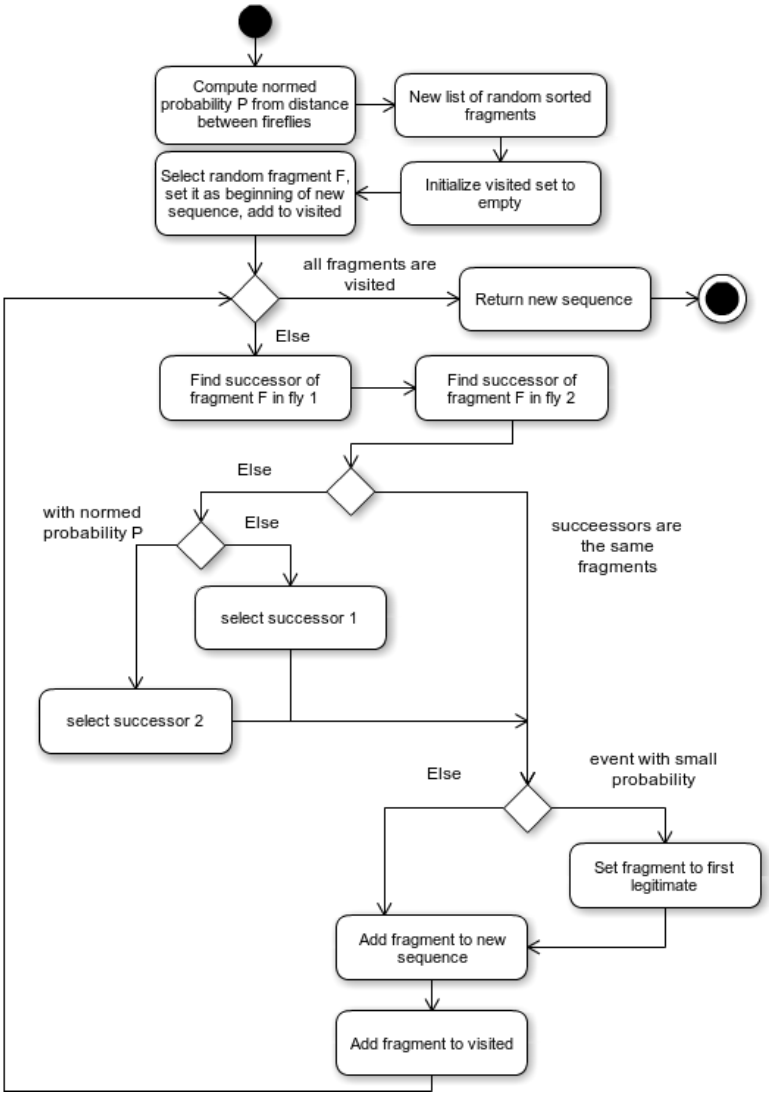

Figure 1. Activity diagram for *SCM* movement

We are completely satisfied with original proposed mathematic model in equation 5.1. The firefly quality, or better its light intensity in distance zero is the sum of overlaps over all fragments in a permutation. Another interesting model from [6] is also based on overlaps but penalizes permutations, where two good overlaps are far from each other.

Selection:
We use three methods. The first one chooses best solutions only from new generated fireflies. The second one chooses best solutions from a union of sets of old and new fireflies. The third one takes constant number of elits from old iteration and combines them with the new fireflies. For quality of solution we use the same function as for attractiveness.

6. Complexity

6.0.1. Data complexity

The main source of data complexity is the table of distances/overlaps between fragments. Size of this table is $N * N$, where N is number of instances/fragments. Data complexity of algorithm is

$$O(N^2). \tag{6.1}$$

Proposed data complexity is specific for many BIA approaches and is one of the main limitations for solving DNA assembly of real organisms size.

6.0.2. Time complexity

The firefly algorithm is nondeterministic. It runs in iterations, until condition is not met, I. In each iteration, for each firefly P, new fireflies are generated. Most computationally intensive part of the algorithm is computation of quality of firefly. It needs to sum $N - 1$ of paths, the complexity is $O(N)$. Chosen data structure used for paths storage also influences the overall complexity. Taking all this into account, the overall complexity is

$$O(I * F * P * N), \tag{6.2}$$

where: I - number of instances, F - number of fireflies in generation, P - number of moves, N - number of fragments.

7. Experiments and results

In the first phase the experiments were performed on freely available Asymmetric Travelling Salesman (ATSP) instances*, instances of TSP where distance from city A to B is not equal to distance from city B to A. The solution of the problem is to find shortest path through all the cities.

The second phase was tested on DNA data. We used publicly available instances of benchmark problems generated by *GenFrag* and *DNAgen* tools. The dataset is available from †. In [13] authors collected often used problem instances, documented them and compared results of various algorithms tested on them. The dataset became standard for testing metaheuristic assembler methods. This is the reason, why we also decided to use these problem instances.

In Table 1 we can see the statistics obtained from algorithm runs on available problem instances. The algorithm was repeatedly running 10 times for every instance. The represented value is an overall sum of overlaps in permutation. We monitored the best and the worst result, variance, mean and median. Instances *bx842596* have high variance compared to *acin*, both are almost the same dimension. The reason could be various values of coverage for these problems.

In Table 2 we can see the performance of our solution compared to other algorithms. For *GenFrag* instances, of smaller dimension, algorithm performed similarly to other methods. It reached the best result in 2 / 7 instances. For instance *x60189_4* it reached optimal value. For instance *x60189_5* it reached best results from compared algorithms. For other 5 instances it performed slightly worse, the furthest from optimum in instance *m15421_7*.

For *DNAgen* instances the results were more interesting. From all compared methods our algorithm reached the best result in 3 from 9 instances. In the other 2, only method *PPSO+DE* was better. In other 3 instances results were slightly worse than others. In the worst instance *j02459_7* the results differed in more than 5 %.

Our algorithm exceeded the expectations, it outperformed compared algorithms in various instances. For others it performed similarly. We successfully adapted firefly algorithm to solve problem of DNA assembly.

*http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/
†http://www.mallen.mx/fragbench/
†http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Instance	Mean	Median	Minimum	Maximum	Variance
x60189_4	11466.1	11487.0	11320.0	11487.0	3486.1
x60189_5	13957.25	13961.5	13841.0	14075.0	9859.9
x60189_6	17885.75	17880.5	17673.0	18097.0	21912.2
x60189_7	20744.0	20718.5	20565.0	20898.0	15334.5
m15421_5	37438.4	37401.0	37026.0	37743.0	51693.8
m15421_6	46617.8	46589.0	46227.0	47033.0	61359.2
m15421_7	51419.5	51408.0	51353.0	51509.0	5643.0
j02459_7	108587.0	108588.5	108470.0	108701.0	9234.0
bx842596_4	211014.25	211013.0	210377.0	211654.0	378990.9
bx842596_4	413045.5	412919.5	412713.0	413630.0	164264.3
acin1	44966.4	44949.0	44827.0	45160.0	10327.1
acin2	147218.8	147236.5	146976.0	147460.0	42134.5
acin3	164306.75	164212.5	164150.0	164652.0	55348.9
acin5	162753.5	162767.0	162565.0	162915.0	21673.6
acin7	179907.5	179908.0	179901.0	179913.0	24.3
acin9	333388.75	333425.0	332890.0	333815.0	217396.9

Table 1. Statistics for *DNAgen* and *GenFrag* instances

8. Conclusion and future work

In this paper we described new approach to DNA assembly based on the firefly algorithm. The algorithm, essentially a metaheuristic, shows interesting optimization possibilities. We designed new algorithm operators and evaluated them on benchmark *DNAgen* and *GenFrag instances*.

We run multiple experiment types to explore influence of algorithm parameters on the quality of results. Finally, we generated solutions for all available DNA benchmark problems.

Based on these results we can say that firefly algorithm is comparable in performance with other BIA approaches. For larger *DNAgen* instances it even outperformed the algorithms used for comparison. One result of experiments is that firefly algorithm can be successfully adapted to solve the problem of DNA fragment assembly.

Fine tuning and optimizing the implementation of our prototype is likely to bring improvements. Further research e.g. on employing a combination of new operators could also amend the proposed solution.

Benchmark	LKH [14]	PPSO [13]	QEGA [2]	SA [2]	PALS [15]	SAX [15]	**FF**
GenFrag instances							
x60189 4	11478	11478	11476	11478	11478	11478	**11487**
x60189 5	14161	13642	14027	14027	14021	14027	**14075**
x60189 6	18301	18301	18266	18301	18301	18301	**18097**
x60189 7	21271	20921	21208	21271	21210	21268	**20898**
m15421 5	38746	38686	38578	38583	38526	38726	**37743**
m15421 6	48052	47669	47882	48048	48048	48048	**47033**
m15421 7	55171	54891	55020	55048	55067	55072	**51509**
DNAgen instances							
j02459 7	116700	114381	116222	116257	115320	115301	**108701**
bx842596 4	227920	224797	227252	226538	225782	223029	**211654**
bx842596 4	445422	429338	443600	436739	438215	417680	**413630**
acin1	47618	47264	47115	46955	46876	46865	**45160**
acin2	151553	147429	144133	144705	144634	144567	**147460**
acin3	167877	163965	156138	156630	156776	155789	**164652**
acin5	163906	161511	144541	146607	146591	145880	**162915**
acin7	180966	180052	155322	157984	158004	157032	**179913**
acin9	344107	335522	322768	324559	325930	314354	**333815**

Table 2. Overall results comparing firefly algorithm with other algorithms [13]

Parallel implementation of the algorithm seems to be a viable option. The firefly algorithm is such that the computationally hard parts require minimum communication.

References

[1] **Ahmed, Z.,** Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, *International Journal of Biometrics*, **3** (6) (2010), 96–105.

[2] **Firoz, J.S., M.S. Rahman, and T.K. Saha,** Bee algorithms for solving DNA fragment assembly problem with noisy and noiseless data, *Proceedings of the Fourteenth International Conference on Genetic and evolutionary computation GECCO'12, New York, NY, USA,* ACM, 2012, 201–208.

[3] **Indumathy, R. and S.U. Maheswari,** Nature inspired algorithms to solve DNA fragment assembly problem: A survey, *International Journal of Bioinformatics Research and Applications*, **2** (2) (2012), 45–50.

[4] **Jati, G. and Suyanto,** Evolutionary discrete firefly algorithm for travelling salesman problem, *Adaptive and Intelligent Systems*, ed. A. Bouchachia, LNCS **6943**, Springer Berlin Heidelberg, 2011, 393–403.

[5] **Li, L. and S. Khuri,** A comparison of DNA fragment assembly algorithms, *Proc. of the Int. Conf. on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, CSREA Press, 2004, 329–335.

[6] **Luque, G. and E. Alba,** Metaheuristics for the DNA fragment assembly problem, *International Journal of Computational Intelligence Research*, **1** (2) (2005), 98–108.

[7] **Meksangsouy, P. and N. Chaiyaratana,** DNA fragment assembly using an ant colony system algorithm, *The 2003 Congress on Evolutionary Computation, 2003. CEC'03*, 2003, vol. 3, no. C, 1756–1763.

[8] **Miller, J.R., S. Koren, and G. Sutton,** Assembly algorithms for next-generation sequencing data, *Genomics*, **95** (6) (2010), 315–27.

[9] **Pevzner, P.a. and H. Tang,** Fragment assembly with double-barreled data, *Bioinformatics (Oxford, England)*, **17** (1) (2001), S225–33.

[10] **Pevzner, P.a., H. Tang, and M.S. Waterman,** An Eulerian path approach to DNA fragment assembly, *Proc. of the National Academy of Sciences of the United States of America*, **98** (17) (2001), 9748–53.

[11] **Schatz, M.C., A.L. Delcher, and S.L. Salzberg,** Assembly of large genomes using second-generation sequencing, *Genome research*, **20** (9) (2010), 1165–73.

[12] **Yang, X.S.,** Firefly algorithms for multimodal optimization, *Proc. 5th Int. Conf. on Stochastic Algorithms: Foundations and Applications SAGA'09,* Springer, Berlin, Heidelberg, 2009, 169–178.

[13] **Mallén-Fullerton, G.M., J.A. Hughes, S. Houghten, and G. Fernández-Anaya,** Benchmark datasets for the DNA fragment assembly problem, *Int. J. Bio-Inspired Comput.*, **5** (6) (2013), 384–394.

[14] **Helsgaun, K.,** An effective implementation of the lin-kernighan traveling salesman heuristic, *European Journal of Operational Research*, **126** (2000), 106–130.

[15] **Minetti, G. and E. Alba,** Metaheuristic assemblers of DNA strands: Noiseless and noisy cases, *2010 IEEE Congress on Evolutionary Computation (CEC),* 1-8.

RNDr. Anna Bou Ezzeddine, PhD.
Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Bratislava, Slovakia
anna.bou.ezzeddine@stuba.sk

Bc. Stefan Kasala
Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Bratislava, Slovakia
stefan.kasala@gmail.com

prof. Ing. Pavol Navrat, PhD.
Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Bratislava, Slovakia
pavol.navrat@stuba.sk