American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-12, Issue-5, pp-135-144
www.ajer.org
Research Paper Open Access

Discrete Cat Swarm Optimization Algorithm for Solving DNA Fragment Assembly Problem: Performance Analysis

ASMAE YASSINE*[1],MOHAMMED ESSAID RIFFI[1], MORAD BOUZIDI[1]
[1](Department of Computer Science, Chouaib Doukkali University, Morocco
Corresponding Author: Asmae Yassine
Corresponding Author E-mail: yassine.asmae1@gmail.com

ABSTRACT: *Nowadays the demand for DNA sequencing is growing exponentially due to the need in tracing viral genome and its evolution within the host besides helping in the study of genetic variation of diseases and developing their treatment with the precision medicine. Because the DNA sequence cannot be interpreted in one step with current sequencing technology, the genome is randomly broken to small fragments and then linked together using the DNA assembly methods to reconstruct the original structure of DNA. Thus since assembling DNA fragments in correct sequence is a hard NP problem, a need arises to construct efficient metaheuristics based algorithms for solving this combinatorial optimization problem. This work present DCSO a novel discrete swarm metaheuristic as an assembling algorithm based on Cats behavior with a comparative study which is a novelty in the DNA fragment assembly problem. The proposed assembler is designed to help finding the information from short sequence fragments by assembling them into contiguous DNA sequences by searching the longest possible overlap between any two sequences. In this work, the Discrete Cat Swarm Optimization Algorithm have been designed, implemented and tested with appropriated parameter settings and also compared with common used algorithms in the literature. The experimental results have shown the high performance and efficiency of our proposed assembling algorithm.*

KEYWORDS *Bioinformatics; Combinatorial Optimization; Metaheuristics; Swarm Algorithm; Fragment Assembly*

Date of Submission: 10-05-2023 Date of acceptance: 21-05-2023

I. INTRODUCTION

Swarm intelligence is a term introduced by Gerardo Beni and Jing Wang in the context of cellular robotic systems [1] that describes the natural behavior manifested by group of birds, ants or fish when they move together in perfect unison. This motion is called ”swarm behavior” which can tell us the interaction between animals of same diversity species type for moving together or finding food and also escaping predators.

Many algorithms have adapted Swarm intelligence] which were applied in many fields like genetics [2], medical data mining [3], logistics [4], robotics [5], etc.

In genetics, in aim to solve the DNA fragment assembly problem, the authors have adapted in [6] the real version of the particle swarm optimization algorithm (PSO) despite that the DNA sequence assembly problem is a discrete optimization problem. The authors have adapted the PSO algorithm to discrete form using shortest position value (SPV) rule. PSO is built on the premise that every individual is a potential solution to a problem. Every element of the PSO represents the order in which DNA fragments are aligned to produce a consensus sequence in the case of DNA sequence assembly problems. For the DNA sequence assembly problem, each individual has a dimension value that is equal to the number of fragments used in the assembly.

A memetic PSO algorithm was proposed in [7] consisting of the combination of tabu search and simulated annealing-based variable neighborhood search local search VNS that were used for enhancing the quality of the results. Another research [8] presented a memetic Gravitation Search Algorithm (MGSA) algorithm for solving DFA problems. Using SPV rule, the algorithm converts continuous position values into discrete sequences, initializes the population by a tabu search, and adopts simulated annealing with the use of VNS as the local search method

A four-phase approach based on rigorous design criteria is presented in [9] for DNA fragment assembly, it have used a limited form of multiple sequence alignment to detect and frequently correct errors in data. While the authors of this paper [10] built and evaluated an exact method (exFRAG [11]) and four heuristic algorithms (gaFRAG [12], mwcFRAG, mFRAG, and bbFRAG) for the assembly of DNA fragments, the exFRAG inspired by Elloumi [11] consists of three steps: The first involves using the dynamic semiglobal programming approach to remove redundant sequences. The path from the root node to the leaf with highest weight is the optimal fragment order, which is used to form a weighted branch and a linked tree. The third step involves creating the layout for a better combination of the fragments and using majority rule to determine the original sequence. GaFRAG inspired by Parsons et al. [12], is a hybrid genetic algorithm that uses pair overlap and genetic algorithm operators to order the fragments before gradually constructing the layout using a sorted order representation and two crossover points. It was the first heuristic used in their work. The program mwcFRAG creates contigs with the highest weight, utilizes the greedy method to locate Hamiltonian paths, then employs a progressive method to arrange the fragments and create the layout. The CAP2 algorithm served as the model for mFRAG, which has three phases like other algorithms. The semi-global dynamic programming approach serves to pair-align the fragments in the first phase, like gaFRAG. The contigs are then combined in the second phase using a greedy approach. In order of decreasing fragment order, the contigs are then combined. The layout is gradually constructed in the final step to build the original DNA sequence [10].

Recently, the authors have introduced the Recentering-Restarting Genetic Algorithm (RRGA) in [13] and Recentering–Restarting Hybrid Genetic Algorithm (RRHGA) [14] using of multiple variants of Genetic Algorithm. It's stated in the paper [13] that one of the RRGA's key advantages is its capacity to avoid becoming fixated on local optima. This is because of how the RRGA traverses the search space and the peculiarities of the necessary dynamic representation. A center or reference point, which is a precise representation of a potential solution to the problem, must be chosen before the algorithm starts. This center may be seeded or chosen at random. Uzma and Halim [14] see that it is better to start with a solid solution while the RRGA refines prospective solutions. The population is produced after the center is chosen. In the direct representation, each member of the population is created by applying a string of n transpositions to the population's center; evolution will change the ordered lists of fragments as is stated in the paper [14]. As an evolutionary operator, PALS have been used with the Restarting and Recentering Genetic Algorithm (RRGA) in [14]. Using overlap scores and the quantity of contigs, the effectiveness of the new proposal is measured. The two primary goals of the proposal's bi-objective optimization problem are to maximize the sum of overlap score and reduce the number of contigs. According to authors, the initial representation of the fragments' order used to run the RRGA is referred to as center. The dataset's default arrangement for the pieces was represented by the center. Then, an 2-opt heuristic was used to optimize the center. The representation processes was used to produce a set of chromosomes after the center has been generated. Based on fitness value, the best chromosome was chosen from the group of chromosomes. Based on the fitness value, a comparison between the best chromosome and the center was made. The numbers of transpositions have been reduced by 5 percent and the center was exchanged with the better chromosome when the fitness value of the best chromosome exceeded the center [14]. The authors have used three different sorts of experiments to evaluate their work and also compared its results with different algorithms The RRHGA have shown its good performance over the three algorithms.

In our work, we propose the metaheuristic algorithm Discrete Cat swarm optimization algorithm (DCSO) known for its efficiency in many continuous optimization problems. In our case we studied the application of the adaptative discrete metaheuristic in genetic domain for its big importance in real life for optimizing the genetic assembler which is also a discrete problem. In this study we compare our implementation with other algorithms: Recentering–Restarting Hybrid Genetic Algorithm (RRHGA), Genetic Algorithm (GA), Descent Algorithm and Particle Swarm Optimization Algorithm (PSO).

In the second section of this research paper, we describe the DNA Fragment Assembly Problem, the third present the DCSO algorithm, the fourth is about the adaptation of the DCSO algorithm in FAP. The section 5 presents the comparison analysis of the test results with other algorithms by using instances of Genfrag with the variation of the parameters settings of the algorithm. Finally, last section concludes and gives some perspectives about future work.

II. FRAGMENT ASSEMBLY PROBLEM

The DNA sequence is composed by 4 types of nucleotides forming the double-helix which are Adenine (A), Thymine (T), Guanine (G), and Cytosine (C). To determine the DNA sequence of an organism's genome Shotgun sequencing is used. After the amplification process (making many copies of the original DNA sequence), the Shotgun method aim to break the long DNA sequences into small reads for sequencing them individually and then reassembling them. DNA fragments assembly problem is a challenging process it refers to aligning and putting a large number of short DNA fragments of a large genome which are individually sequenced in order to reconstruct the original DNA sequence.

The Overlap-Layout-Consensus (OLC) method, Bruijn graphs, and greedy graph-based algorithms are the three primary approaches that are frequently employed to overcome this issue. In our scenario, shotgun sequencing is used to reproduce the DNA sequence. The OLC model in fig.1 serves as the foundation for this process; this is accomplished using the following steps:

- Overlap: The initial stage entails knowing the overlaps between all the fragments. All approximate overlaps between fragments are determined in this step. In order to achieve this, every pair of fragments is compared in order to determine which pair of fragments has the greatest matching overlap. Programming algorithms applied to semi-global alignments like the Smith-Waterman algorithm [15] are primarily used for this.
- Layout: The goal of this stage which is the main element of any OLC DNA fragment assembler is to determine the proper placement of each fragment by locating the fragments' sequential order that maximizes the overlap score of all preceding fragments. It's proven that this phase is an NP-hard problem [16].
- Consensus: After constructing the layout, this last phase entails reconstructing the entire DNA

. The Overlap Layout Consensus model is illustrated in fig.1 from the fragmentation of the copies of the original DNA Sequence to the reconstruction of the consensus.

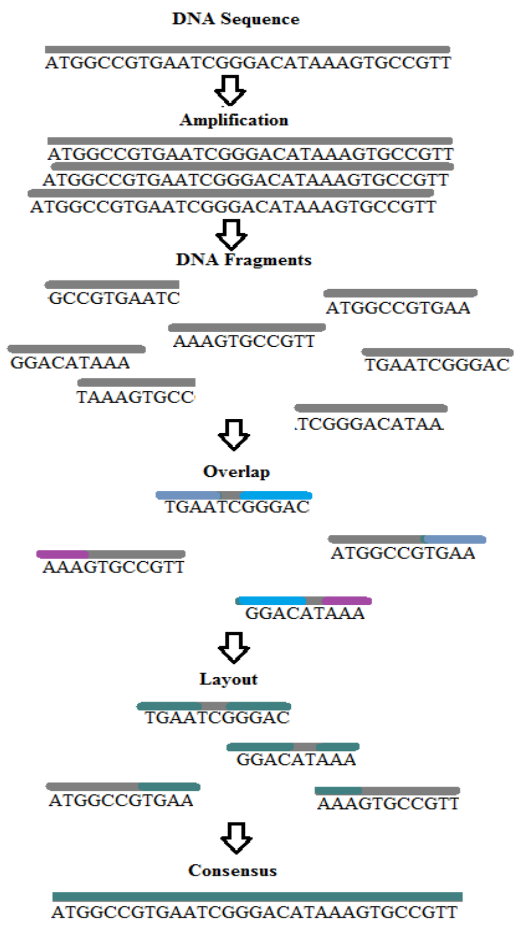

Fig.1. Overlap-Layout-Consensus steps.

In order to comprehend the issue more fully, we must be familiar with the following fundamental terms:

- Fragment: A brief DNA sequence up to 1000 bps in length.

- Shotgun data: A collection of fragments.

- Prefix: The first n characters of a fragment's substring.

- Suffix : Substring of the final n characters of a fragment f.

III. DISCRETE CAT SWARM OPTIMIZATION ALGORITHM

Cat swarm optimization in a swarm intelligence algorithm, adopting the natural behavior of cat, introduced in 2008 by Chu, Tsai and Pan [17]. The cats are lazy creatures that spend most of the time resting (Seeking mode) but while rest they are aware about the entourage and when they feel a target they start moving toward them (Tracing mode). Each cat is represented by its position (the solution), velocity and flag (Seeking mode or Tracing Mode). The mixing ratio (MR) determines in which mode the cat is going to be.

3.1 Tracing mode

Tracing mode imitates the tracing behavior of cats while observing its prey. Once a cat goes into tracing mode, random velocity values are given for every dimension of a cat's position. Moving cats in tracing mode can be described in 3 steps as follow:

1) Velocities (Vk,d) are updated for all the dimensions according to the equation (1) where (Xk,d) is the cat's position and (Xbest,d) is the cat's best solution and position having the best fitness value, r1 has random values the interval of [0,1] and c1 is a constant.
2) Check if the velocities value exceeds the maximum value if so, then it takes the maximum velocity.
3) The Cat k position is updated according to the following equation (2).

$$V_{k,d} = wV_{k,d} + r_1c_1(X_{best,d} - X_{k,d}) \quad (1)$$

$$X_{k,d} = X_{k,d} + V_{k,d} \quad (2)$$

3.2 Seeking mode

Searching mode simulates a cat's resting behavior and uses and adjusts four basic parameters:

- CDC: Counts of dimension to change.
- SMP : Seeking memory pool.
- SPC : Self-position considering.
- SRD: Seeking range of the selected dimension.

Searching mode is presented as follows:

1) Copy SMP copies of the ith cat in the seeking mode or make SMP-1 copies of ith cat if the SPC value is true
2) Choose for each copy random CDC dimensions to be mutated and generate the SRD value.
3) Evaluate the fitness value (FS) for each candidate positions. Then select the cat's next position where higher FS candidate points.
4) Perform the mutation and change the present position.
5)

The complete mode DCSO Fig.2 combines the seeking mode and the tracing mode using the mixing ratio (MR), which will decide in which mode each cat will move to. The steps of the DCSO's complete mode are as follow:

1) Specify the solution set's upper and lower bounds.
2) Create random N Cats with speed value less than or equal to set maximum speed value predefined. The position xi of each cat is randomly generated and initialized.
3) Initialize the flag (SM or TM) of each cat according to MR.
4) Update the value of the flag of each cat according to MR.
5) The cat moves with respecting its mode indicated by its flag
6) Evaluate each cat's fitness, and saving best cat x_{best} into memory.
7) Check the termination condition. If yes, terminate the program. If not, repeat 4), 5) and 6).

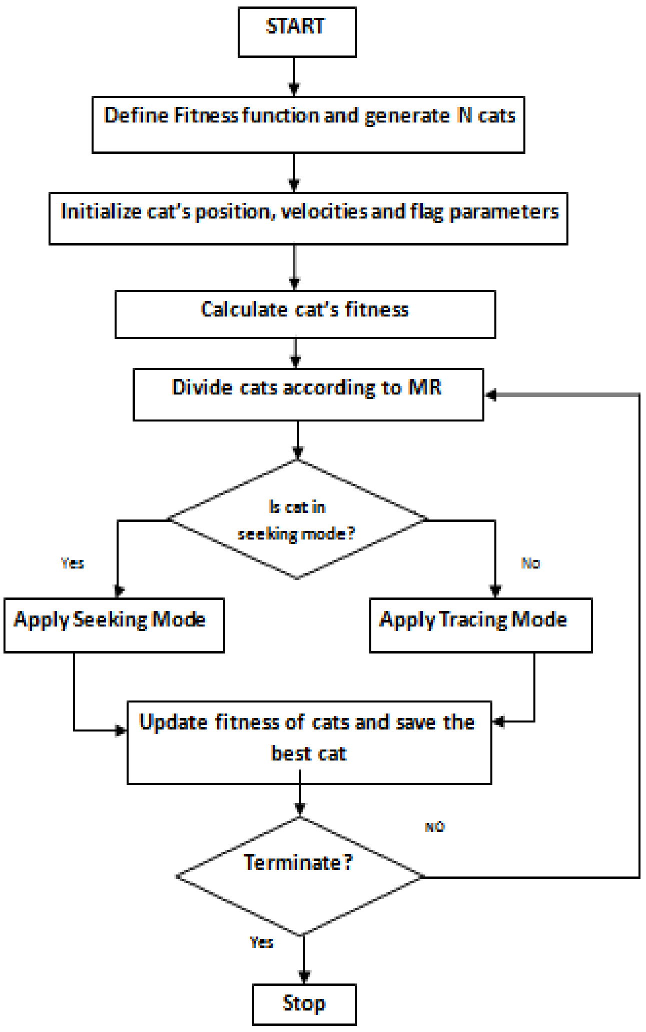
START
Define Fitness function and generate N cats
Initialize cat's position, velocities and flag parameters
Calculate cat's fitness
Divide cats according to MR
Is cat in seeking mode?
Yes
No
Apply Seeking Mode
Apply Tracing Mode
Update fitness of cats and save the best cat
NO
Terminate?
Yes
Stop

Fig. 2. Flowchart Of the Discrete Cat Swarm Optimization Algorithm

IV. SOLVING DNA FRAGMENT ASSEMBLY PROBLEM WITH DCSO

The DNA fragment assembly problem is known as a combinatorial optimization problem. To resolve this problem our proposed assembler DCSO aim to find the consensus DNA sequence by assembling the short fragments to find the original DNA sequence the adaptation of the algorithm as presented in fig.3.

4.1 Solution representation

In our adaptation of DCSO, each cat is characterized by its position that represents the solution to the problem and also by its velocity and flag that decide whether the cat is in tracing mode or in searching mode. Solutions are known as N permutations. The fragment represents an element of the cat's positions, the permutation presents a fragment and its index presents its position in the solution.

In case of DNA fragment assembly problem the optimal solution is maximizing overlaps between neighboring fragments order. First the fragments must be aligned according to the fragment order then longest overlap length fitness between a fragment's suffix and another fragment's prefix is calculated. Fragments scores are computed by counting the fragment's matching nucleotide. Matching fragment's scores are calculated using Eq (3).

$$\text{Score}_{i,j} = \begin{cases} 0, \text{if nucleotid does not matches} \\ \text{Score } i, i+1, \text{otherwise} \end{cases} \qquad (3)$$

$$\max \text{Fi}(x) = \sum_{j=o}^{D-1} \text{Scorej}, j+1 \qquad (4)$$

In eq(3), $\text{Score}_{i,i+1}$ is a matching overlap score of two arranged successive fragments of sequence vector where i and i+1 are the index in a set (1, 2, . . . , n) representing the sequence vector. After calculating the score of pair fragment total score is calculated in DSCO by using the fitness function: Eq (4).

```
Create N cats;
Initialize the cat's position, velocity, and flag parameters;
Repeat;
Based in MR reinitialize the flag of each cat;
for each cat k in the swarm do
    if flag of cat k is TM then
        V k = w*V k+ r * c * (X best - X k) (1);
        X k = Xk + Vk (2);
    else
        make j copies of cat k;
        if SPC==true then
            consider cat k as a candidate;
            j=SMP-1;
        else
            j= SMP;
        end
        for each copy do
            Select a number of dimensions to be mutated based on CDC;
            Evaluate the fitness of each copy;
        end
        for each cat do
            Randomly select a new position for cat k;
        end
    end
    Update the fitness of each cat;
    Save the best position of the cat in the memory;
end
Until (Condition is satisfied);
```

Fig. 3. Pseudo code of Discrete Cat Swarm Optimization Algorithm for DNA Fragment Assembly Problem

V. EXPERIMENTS AND RESULTS

In this section, we present our experimental setup and the results obtained by our algorithm. First, we show the features of the selected DNA-FAP instances. Next, we evaluate the DCSO algorithm with a comparison analysis with RRHGA, GA, Descent and PSO and with a comparison of the parameter settings used in the tests. Then we analyze and discuss the obtained results.

5.1 Experimental Setup

The proposed algorithm and the other methods have been programmed with C++ language The experiments were conducted in Windows 7 running on an Intel core i7 Processor with 4GB RAM and has been tested on some artificial fragments in GenFrag datasets X60189: A cluster of fibronectin type III repeats 3835 bases long, found in the human major histocompatibility complex class III region [18]. We give a summary on the different features of the datasets in Table 1.

Table 1: Information of Datasets for DNA-FAP problem.

Instances	Coverage	Average length	Fragments	Sequence length
$X60189_4$	4	395	39	3835
$X60189_5$	5	286	48	3835
$X60189_6$	6	343	66	3835
$X60189_7$	7	387	68	3835

Table 2: Parameter Settings of DCSO.

Parameter	Value or range
SMP	Number of fragments
CDC	5
MR	20%
W	0.7
C1	2
R1	[0,1]
Dimension	Number of fragments
Population Size	20
Minimum iterations	8000

5.2 Result and analysis

Table 2 summarizes the different parameters settings used in the DCSO algorithm. Table 3 presents the different results of DCSO algorithm with parameters variations for instance X601894. Table 4 shows the numerical results of DCSO approach with other methods RRHGA , GA, Descent and PSO-SPV for the instances of the X60189 cluster. The first column presents the name of each instance. The best fitness results obtained by using the DCSO algorithm are presented in second column and the best fitness values ever obtained by the algorithm RRHGA collected from [14] which is a recent research, and our implemented GA, Descent and PSO-SPV are also indicated in the following columns in the table. Bold font indicates best result obtained in each experiment for analysis. The synthesis of the results of the DCSO, RRHGA and PSO-SPV algorithms is visualized in the Bar-Plot in fig.4.

Table 3: Comparison of the results obtained with DCSO, RRHGA, GA, Descent and PSO algorithms on different instances

Sequence Instances	DCSO Algorithm	RRHGA[14]	GA	Descent Algorithm	PSO-SPV Algorithm
X60189$_4$	**10821**	6488	2823	5645	5408
X60189$_5$	**13148**	8655	2671	7506	6331
X60189$_6$	**17162**	9943	4177	8523	6837
X60189$_7$	**19288**	11546	4870	9085	8667

Table 4: Comparison performance of DCSO

Iterations number For instance X60189$_4$	Fitness with MRatio=0,2		Fitness with MRatio=0,45		Fitness with MRatio=0,5	
	Population size=20	Population size=40	Population size=20	Population size=40	Population size=20	Population size=40
Iter=3000	9848	9579	**10435**	6163	9801	5825
Iter=8000	10029	9620	**10682**	6163	10204	5825
Iterations number For instance X60189$_5$	Fitness with MRatio=0,2		Fitness with MRatio=0,3		Fitness with MRatio=0,4	
	Population size=30	Population size=40	Population size=30	Population size=40	Population size=30	Population size=40
Iter=3000	**12880**	12878	12542	11879	11931	12410
Iter=8000	**13117**	12941	12724	12081	12716	12586
Iterations number For instance X60189$_6$	Fitness with MRatio=0,2		Fitness with MRatio=0,25		Fitness with MRatio=0,5	
	Population size=40	Population size=50	Population size=40	Population size=50	Population size=40	Population size=50
Iter=3000	7250	15434	6669	**15897**	14133	14885
Iter=8000	7250	15713	6669	**15958**	14836	15122
Iterations number For instance X60189$_7$	Fitness with MRatio=0,5		Fitness with MRatio=0,6		Fitness with MRatio=0,7	
	Population size=60	Population size=80	Population size=60	Population size=80	Population size=60	Population size=80
Iter=3000	17297	8969	**17531**	8707	17053	9391
Iter=8000	17537	8969	**18043**	8707	17813	9391

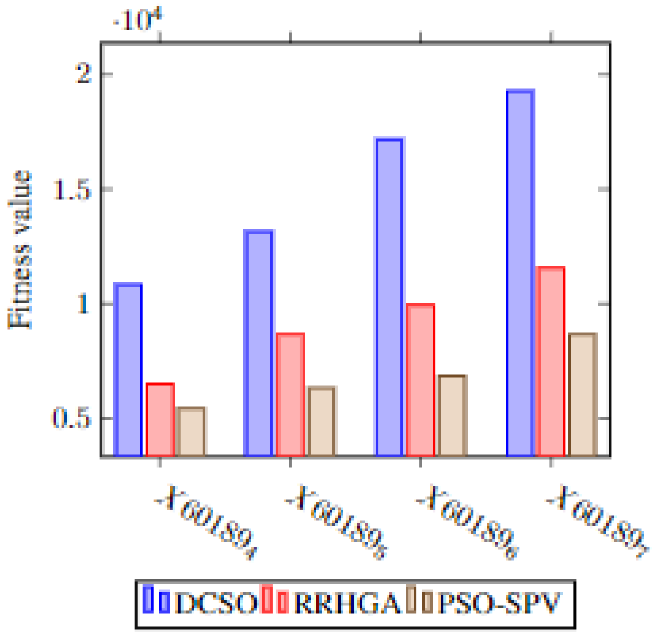

Fig. 4. Performance comparison between DCSO, RRHGA and PSO-SPV on the different instances

5.3 Discussion

From comparing the results of best fitness of each algorithm presented in the Table 3 we can see: the discrete cat swarm optimization algorithm have led to very accurate solutions for the DNA fragment assembly problem and we can see that it also performs much better comparing to the other algorithms: the DSCO algorithm in all of the considered instances outperforms the rest of the algorithms GA, Descent and PSO-SPV with significance results.

It's clear seen from the Table 4 that several parameters have affected on obtained results: MRatio have been varied from 0,2 to 0,5 depending to the instance to best results. The population number was adapted too from 20, 30, 50 and 60 for X60189(4), X60189(5), X60189(6) and X60189(7) respectively showing better score of fitness value. From the same Table 4 we can observe that the variation of size of population change make important changes in the final results for the X60189(4) instance for example the size of population equal to 20,30 and 40 had given 10029, 5774 and 9620 respectively for same number of iterations (8000) and MRatio value(0,2).

The process is executed for this number of iterations due to short computational time of DCSO, Each number of iterations and instance have been executed with its appropriate parameter settings although it gives reasonable changes, the general parameter settings presented in Table 2 applied for all the instances give automatically high fitness scores values. Thus, the parameter settings play a major role in the convergence rate and it produces quality solutions in the DCSO algorithm for the Genfrag data instances.

The number of iterations also was influencing on the fitness value as presented with the line plots in fig.5. As the number of iterations executed increase, the obtained score while running the DCSO algorithm is also increased.

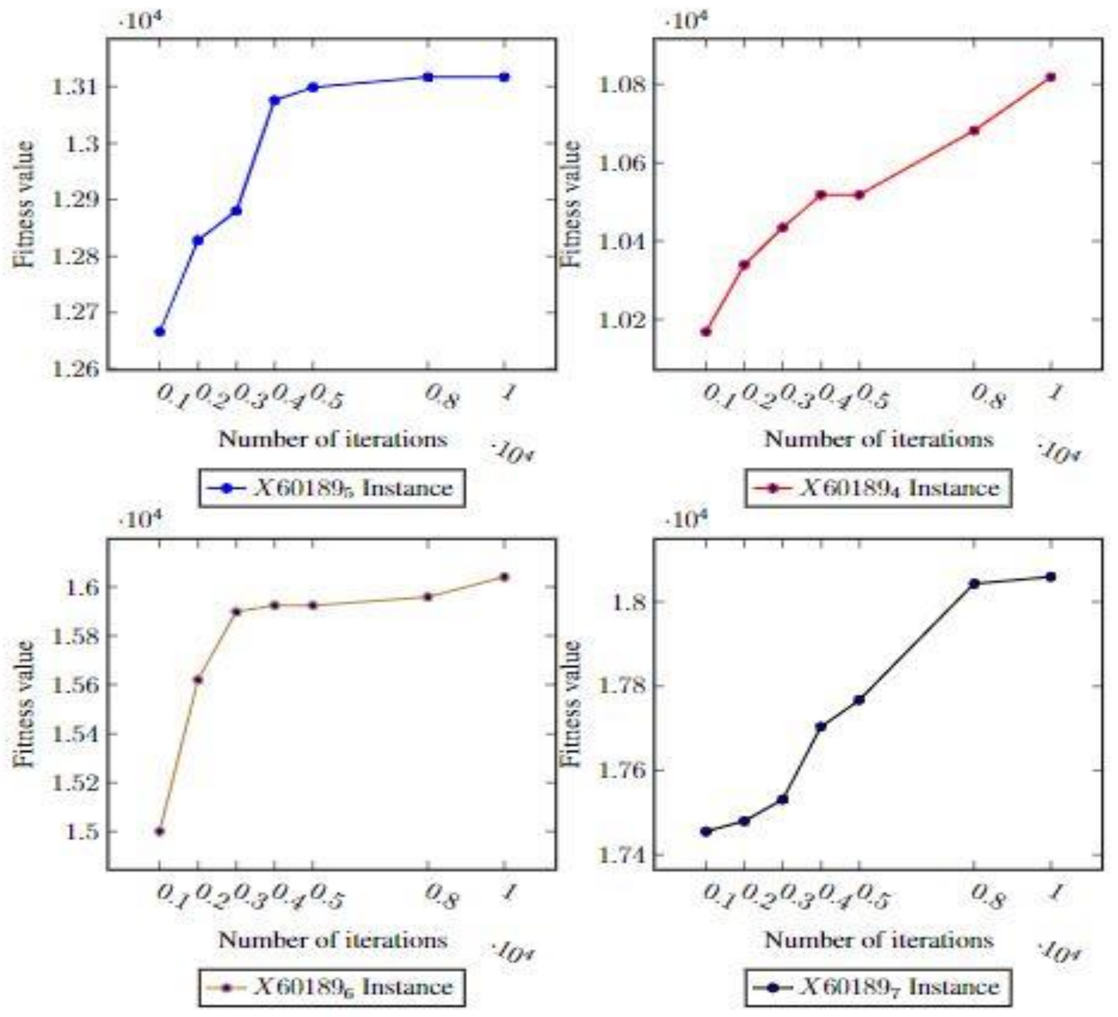

Fig. 5. Comparison graph for DCSO algorithm's best fitness values with number of iterations variation on different instances

VI. CONCLUSION

In this article we presented the application of the novel algorithm Discrete Cat Swarm Optimization for solving the DNA Fragment Assembly Problem and we presented a comparative study to analyze its performance with other three implemented algorithms and with a recent research algorithm. We have adjusted the parameters of DCSO to obtain the best configuration of the algorithm for this problem. The results obtained by testing on Genfrag instances have demonstrated the good performance of this algorithm over other algorithms. In the future we aim to apply various nature inspired algorithms for DNA sequence assembly problem and to develop hybrid local search methods for the algorithm to reach better fitness score and compare the results with our proposed algorithm.

REFERENCES

[1]. A. Garg, and P. Gilland and P. Rathi and Dr K. K Garg. An Insight into Swarm Intelligence. International Journal of Recent Trends in Engineering.(2009)
[2]. Nawzat Ahmed. Swarm intelligence algorithms in gene selection profile based on classification of microarray data: A review. Journal of Applied Science and Technology Trends. 2:1–09 , 02(2021)
[3]. Mostafa Ghannad-Rezaie, Hamid Soltanian Zadeh, M.-R Siadat, and Kost Elisevich. Medical data mining using particle swarm optimization for temporal lobe epilepsy, pages 761 – 768. 01 (2006)
[4]. Ying Cui. Application of particle swarm optimization algorithm in port ship logistics. Journal of Coastal Research, 115:226.08 (2020)
[5]. Taylan Das, Lale D ulger, and G ulesin Das . Robotic applications with particle swarm optimization (pso). pages 160-165. 05 (2013)
[6]. Verma, Ravi & Singh, Vikas & Kumar, Sanjay. DNA Sequence Assembly using Particle Swarm Optimization. International Journal of Computer Applications. 28. 10.5120/3425-4777. (2011).

[7]. Huang, Ko-Wei; Chen, Jui-Le; Yang, Chu-Sing; Tsai, Chun-Wei. A memetic particle swarm optimization algorithm for solving the DNA fragment assembly problem. Neural Computing and Applications, 26(3), 495–506. (2015).

[8]. Huang, Ko-Wei; Chen, Jui-Le; Yang, Chu-Sing; Tsai, Chun-Wei .A memetic gravitation search algorithm for solving DNA fragment assembly problems. Journal of Intelligent & Fuzzy Systems, 30(4), 2245–2255(2016).

[9]. J. Kececioglu and E. Myers, “Combinatorial algorithms for dna sequence assembly,” Algorithmica, vol. 13, 02. pp.7–51 (1995)

[10]. Y. Jing and S. Khuri, “Exact and heuristic algorithms for the dna fragment assembly problem,” pp. 581 – 582. 09 (2003)

[11]. M. Elloumi, “Algorithms for the dna sequence assembly problem,” Proceedings of the World Conference on Systematics, (2000).

[12]. R. Parsons, S. Forrest, and C. Burks, “Genetic algorithms, operators, and dna fragment assembly,” Machine Learning, vol. 21, 06 (1995).

[13]. J.A. Hughes, S. Houghten, D. Ashlock, Restarting and recentering genetic algorithm variations for DNA fragment assembly: The necessity of a multi-strategy approach, Biosystems 150 35–45. (2016)

[14]. Uzma, ; Halim, Zahid. Optimizing the DNA fragment assembly using metaheuristic-based overlap layout consensus approach. Applied Soft Computing, 92(), 106256–. doi:10.1016/j.asoc.2020.106256. (2020)

[15]. Smith, Temple. Waterman MS: Identification of common molecular subsequences.J. Mol. Biol. 147. https://doi.org/10.1016/0022-2836(81)90087-5(1980)

[16]. Pevzner, P. Computational molecular biology: An algorithmic approach. MIT Press.(2000)

[17]. Shu-Chuan Chu, Pei-Wei Tsai, and Jeng-Shyang Pan. Cat swarm optimization. pages 854–858, Page 7 of 8. 01 (2006)

[18]. Michael L. Engle and Christian Burks. GenFrag 2.1: new features for more robust fragment assembly benchmarks. Bioinformatics,. 10(5):567–568 ,09 (1994)