An Intelligent Optimization Model for Fragment Assembly Problem

Osama Maher*

Faculty of Engineering, Fayoum University, Fayoum , Egypt

* Correspondence: eng.osamamaher21@gmail.com

Abstract. In computational molecular biology, the sequencing of Deoxyribonucleic Acid (DNA) is a very challenging problem as it requires rebuilding the original sequence of DNA from a massive number of fragments. This paper introduces an efficient hybridization between Harris Hawks Optimization (HHO) and Problem Aware Local Search (PALS) to be utilized in solving DNA Fragment Assembly Problem (FAP). The efficiency of the proposed hybrid algorithm (PHHO) is compared with the original PALS, Firefly Algorithm (FA), Genetic Algorithm (GA), Artificial Bee Colony (ABC), and GAG50. The experimental results show the efficiency of the proposed algorithm compared to other approaches.

Keywords: Computational Molecular Biology, Problem Aware Local Search, Harris Hawks Optimization, DNA Fragment Assembly Problem, Optimization.

1. Introduction

With the development of Genetics and Bioinformatics research, there is an urgent need for manipulating formidable DeoxyriboNucleic Acid (DNA) sequences. Unfortunately, the existing DNA sequencing technologies allow the read of DNA chains with lengths ranging from 20 to 1000 bases [1]. This range is disappointing compared to the human genome consisting of 3.2 billion bases. As a consequence, the technique of shotgun sequencing is utilized. This sequencing technique clones the same DNA sequence and separates each clone into several shorter fragments. After that, an assembler is used for finding the original DNA chain. There are several greedy-based assemblers used for this proposal like CAP3 [2], Phrap [3], Celera [4], TIGR [5], and STROLL[6], etc. In these greedy-based algorithms, DNA fragments are assembled according to a high level and complex criterion. So they can find high-quality solutions. However, the main disadvantage of such greedy-based assemblers is that they are only efficient for small and medium-scale problems.

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

On the other hand, metaheuristic algorithms are black-box optimizers that can find near-optimal solutions for large scale problems. Consequently, several metaheuristics have been proposed for optimizing the assembling of DNA fragments or Fragment Assembly Problem (FAP). For instance, Firoz et al. [7] proposed two metaheuristics called Artificial Bee Colony (ABC) and Queen Bee Evolution Based on Genetic Algorithm (QEGA) for solving FAP. Both algorithms were applied for handling noisy and noiseless problem instances. Although the prosperity of the proposed algorithms in solving noiseless FAP instances, they had a disappointing performance for noisy FAP instances.

Minetti et al. [8] presented different approaches based on GA for solving FAP, including GA, GA2o50, GA2o100, GAG50, and GAG100. In GA2o50 and GAG50, half of the search agents are created randomly, and the other half are created using a 2-opt heuristic and greedy algorithm. The whole population of GA2o100 and GAG100 is generated using a 2-opt heuristic and greedy algorithm, respectively. The proposed algorithms had a variety of performances with changing the method of crossover. The most appropriate crossover method was cycle and order crossover for the algorithms that depend on the greedy algorithm. Ezzeddine et al. [9] proposed Firefly Algorithm (FA) for handling FAP. Also, the authors added a new operator to FA that proved its efficiency in solving FAP. Allaoui et al. [10] introduced a hybridization between Crow Search Algorithm (CSA) and a local search method for accelerating the search process. The presented hybrid algorithm achieved good results for FAP.

This paper proposed a hybrid Harris Hawks Optimization based on Problem Aware Local Search (PHHO) for Fragment Assembly Problem (FAP). The rest of this paper is organized as follows:

- Section 2: DNA Fragment Assembly Problem (FAP).
- Section 3: Harris Hawks Optimization (HHO).
- Section 4: Proposed Hybridized Algorithm.
- Section 5: Experimental Results.

2. Problem Formulation

DNA Fragment Assembly Problem (FAP) [11] can be defined as determining the best order of DNA fragments that maximize the overlapping score. There are several methodologies for assembling the resultant fragments of the shotgun technique. Overlap-Layout-Consensus (OLC) [12] approach is a prevalent assembling technique that consists of three main phases:

- Overlap phase.
- Layout phase.
- Consensus phase.

In the first phase, all achievable DNA fragments pairs are compared to determine the best match (overlapping score) between their suffix and prefix. After that, the DNA fragments are ordered according to the computed overlapping score. Finally, the consensus phase derives the most likely DNA sequence (or contig) based on the previous phase. The quality of the results of the consensus phase can be measured by computing the coverage.

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

The coverage of a consensus can be defined as the number of existing bases in all fragments over the total length of the original DNA chain, like:

$$\boldsymbol{coverage} = \frac{\sum_{i=1}^{D} \boldsymbol{length\ of\ the\ fragment\ i}}{\boldsymbol{target\ DNA\ chain\ length}} \quad (1)$$

where $\boldsymbol{D}$ is the number of fragments.

Several research studies have proposed Computational Intelligence (CI) techniques or metaheuristic algorithms for supporting the OLC phases. In this paper, we introduce a hybrid metaheuristic algorithm for efficiently solving the problem of finding the best order of DNA fragments in the layout phase.

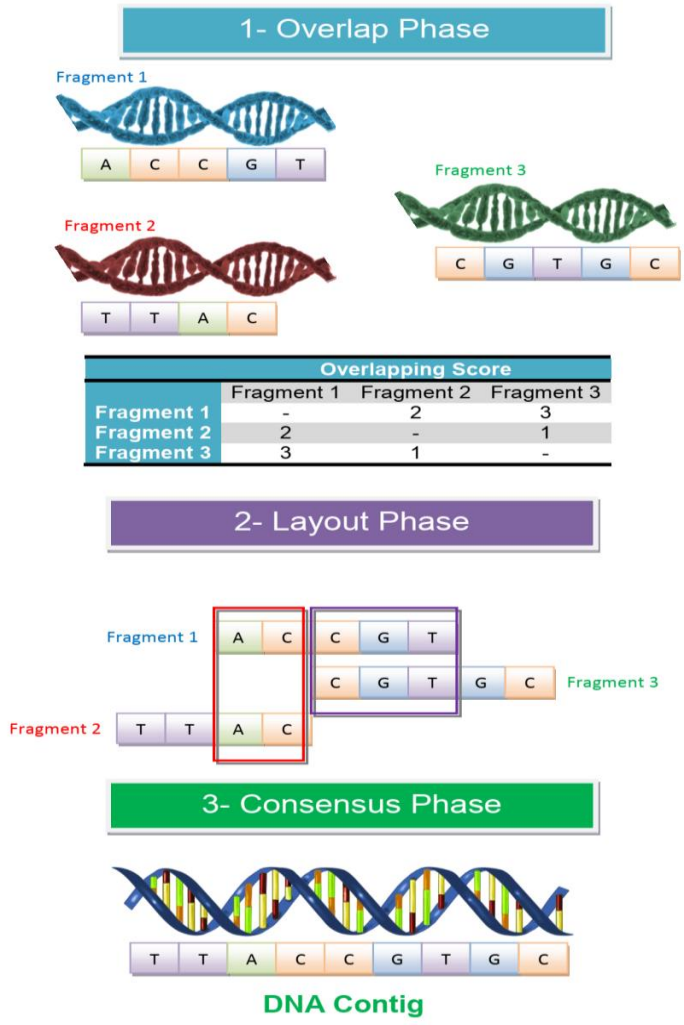

Fig. 1 OLC Approach.

3. Harris Hawks Optimization (HHO)

Harris Hawks Optimizer (HHO) [13] is a CI algorithm that takes its inspiration from the surprise pounce behavior of Harris hawks. Mathematically, this collaborative hunting behavior can be expressed as:

$$x_{t+1} = \begin{cases} x_t^r - r_1 |x_t^r - 2r_2 x_t| & q \geq 0.5 \\ (x_t^* - \bar{x}_t) - r_3\big(lb + r_4(ub - lb)\big) & q < 0.5 \end{cases} \quad (2)$$

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

where x_{t+1} is a new position, x_t^r is a random solution in the current population, x_t^* is the rabbit position at time t. x_t is the hawk position at the time t , r_1, r_2, r_3, r_4 , and q are Uniform random numbers between $(0,1)$, and $\bar{x}_t$ is the average solution of the current population that is calculated as:

$$\bar{x}_t = \frac{1}{N}\sum_{i=1}^{N} x_t^i \quad (3)$$

where N is the total population size and x_t^i is each solution in the current time t.

At each searching iteration, the energy of the rabbit is recomputed in order to balance between the exploration and the exploitation during the search process, as:

$$E = 2E_0\left(1 - \frac{t}{T}\right) \quad (4)$$

where E is the energy of the rabbit, E_0 is the initial energy state, and T is the maximum number of iterations.

The unsuccessful escaping trial of the rabbit causing by two besiege strategies, soft and hard, which are switched according to the escaping probability g. In particular, soft besiege occurs when $g \geq 0.5$ and ≥ 0.5, as follows

$$x_{t+1} = \Delta x_t - E|Jx_t^* - x_t| \quad (5)$$

where J is the rabbit's random jump strength during escaping between $[0,2]$ and Δx_t is the difference between the rabbit location and the current solution at the time t as :

$$\Delta x_t = |x_t^* - x_t| \quad (6)$$

The hard besiege occurs when $g \geq 0.5$ and $E < 0.5$, represented as:

$$x_{t+1} = x_t^* - E|\Delta x_t| \quad (7)$$

The rabbit escapes when $g < 0.5$ during both soft and hard besieges. This makes hawks rapidly dive to detect the rabbit by Levy Flight (LF) pattern, as:

$$x_{t+1} = (\Delta x_t - E|Jx_t^* - x_t|) + \psi \times LF(D) \quad (8)$$

where D is the dimension of the search space, ψ is a vector of random numbers with size $(1, D)$, and LF is a levy flight step that is calculated as:

$$LF = \alpha \times \frac{u \times \sigma}{|\partial|^{\frac{1}{\beta}}}, \sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}}\right) \quad (9)$$

where α is a scaling factor that is set to 0.01, u, ∂ are two uniform random numbers between $(0,1)$, and β is a constant that equals 1.5.

Hence, the hawks decided on the soft besiege strategy according to the fitness value of the generated solution. Algorithm 1. shows the HHO search procedures. Next, the proposed algorithm will be discussed.

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

Algorithm 1 Harris Hawks Optimizer (HHO)

1: Initialize HHO parameters: number of iterations T & hawks N
2: Initialize the population of hawks $X_i(i = 1,2, \dots N)$
3: **while** $(t \leq T)$ **do**
4: Evaluate each hawk position
5 set best solution or rabbit position as x_t^*
6: **for** each hawk
7: $E_0 = 2rand() - 1$
8: $J = 2(1 - rand())$
9: $E = 2E_0\left(1 - \frac{t}{T}\right)$
10: **if** $(|E| \geq 1)$ **then**
11: $q = rand()$
12: **if** $(q \geq 0.5)$ **then**
13: move hawk based on group movement
14: **else**
15: move hawk based on the prey movement
16: **if** $(|E| < 1)$ **then**
17: $g = rand()$
18: **if** $(g \geq 0.5$ & $|E| \geq 0.5)$ **then**
19: soft besieges
20: **else if** $(g \geq 0.5$ & $|E| < 0.5)$ **then**
21: hard besieges
22: **else if** $(g < 0.5$ & $|E| \geq 0.5)$ **then**
23: soft besieges or LF rapid dives
24: **else if** $(g < 0.5$ & $|E| < 0.5)$ **then**
25: hard besieges or LF rapid dives
26: $t++$
27: **end while**
28: **return** best solution x_t^*

4. Proposed Hybridized Algorithm

In this section, we describe the components of the proposed hybrid algorithm.

4.1. Solution Encoding

HHO has achieved a performance for solving continuous optimization problems. To adapt HHO for searching in discrete search space, the candidate solutions are encoded as permutations. There are many methods for matching between the continuous and the discrete search space. This paper utilizes the Smallest Position Value (SPV) methodology [14]. In SPV, the vector elements of the candidate solution are taken by increasing order keys. These keys represent a candidate permutation of DNA fragments of the given problem.

4.2. Objective Function

As discussed before, the fitness of the consensus contig is determined from the sum of the overlapping score of the connected fragments. So, the fitness function of the FAP can be expressed as the sum of the overlapping score of all adjacent fragments pairs in the regarded permutation. This can be calculated as [15]:

$$F(x_i) = \sum_{k=0}^{N-2} w(f_k, f_{k+1}) \quad (10)$$

Where x_i is a candidate solution, and $w(f_k, f_{k+1})$ is the overlapping score of the two adjacent fragments f_k and f_{k+1} .

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

4.3. Problem Aware Local Search

Particularly, using the overlapping score to measure the solution fitness may lead to search stagnation. Therefore, it is more efficient to use the number of resultant DNA contigs as a fitness qualification. i.e. the fittest solution is the solution that has the minimum number of contigs with a target of reaching only one contig. However, the use of the number of contigs can cause overhead time costs. In this chapter, we use an efficient heuristic for leading the searching process of HHO based on the number of contigs instead of the traditional fitness function given in equation (10).

Problem Aware Local Search (PALS) [16] is an efficient heuristic algorithm that is specially designed for the DNA FAP. The main advantage of PALS is the indirect estimation of the number of contigs which mainly reduces the computational cost. This indirect estimation is done by calculating the number of contigs that created or destroyed during the searching process. In addition, PALS considers the overlapping score between fragments for the total qualification of the candidate solution.

The heuristic begins with a single initial permutation of the DNA fragments, as shown in Algorithm 2. Structured movements iteratively modify this permutation. The movement is applied by $applyMovement$ function, which reverses two sub-permutation positions i and j. This movement is selected from the list of all candidate movements $\boldsymbol{L}$ by the function $selectMovement$. Then, PALS mainly depends on the calculation of the variation in the overlap Δ_f and the number of contigs Δ_c between the current solution and the resultant solution after applying the movement. Only the movement that doesn't increase the number of contigs is applied by comparing the overlapping score between two fragments with the $cutoff$ value which, is a predefined threshold value (See Algorithm 3).

Algorithm 2 Problem Aware Local Search

```
Initialize initial solution s
do
    L ← ∅
    for i = 0 to N − 2
        for j = i + 1 to N − 1
            Δc , Δf ← calculateΔ(s, i, j)
            if (Δc < 0) or (Δc = 0 & Δf > 0 ) then
                L ← L ∪ ⟨i, j, Δf, Δc ⟩
        end
    end
    if L ≠ ∅
        ⟨i, j⟩ ← selectMovement(L)
        applyMovement(s, i, j)
while no changes
return s
```


Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

Algorithm 3 $calculate\Delta(\boldsymbol{s}, i, j)$ Function

$\Delta_c \leftarrow 0$, $\Delta_f \leftarrow 0$
$\Delta_f = \Delta_f - w_{s[i-1]s[i]} - w_{s[j]s[j+1]}$
$\Delta_f = w_{s[i-1]s[j]} - w_{s[i]s[j+1]}$
if $w_{s[i-1]s[i]} > cutoff$
  $\Delta_c = \Delta_c + 1$
if $w_{s[j]s[j+1]} > cutoff$
  $\Delta_c = \Delta_c + 1$
if $w_{s[i-1]s[j]} > cutoff$
  $\Delta_c = \Delta_c - 1$
if $w_{s[i]s[j+1]} > cutoff$
  $\Delta_c = \Delta_c - 1$
return (Δ_f, Δ_c)

4.4. PHHO for FAP

This paper hybridizes HHO with PALS (PHHO) to support its searching process with additional exploitation. At each iteration, the generated solution of HHO is sent to PALS to be modified and qualified after being discretized by SPV. Then, this new solution and its fitness value are sent to HHO after being converted to a continuous vector in order to resume the searching process. Fig. 2 shows the flowchart of the proposed hybrid algorithm.

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

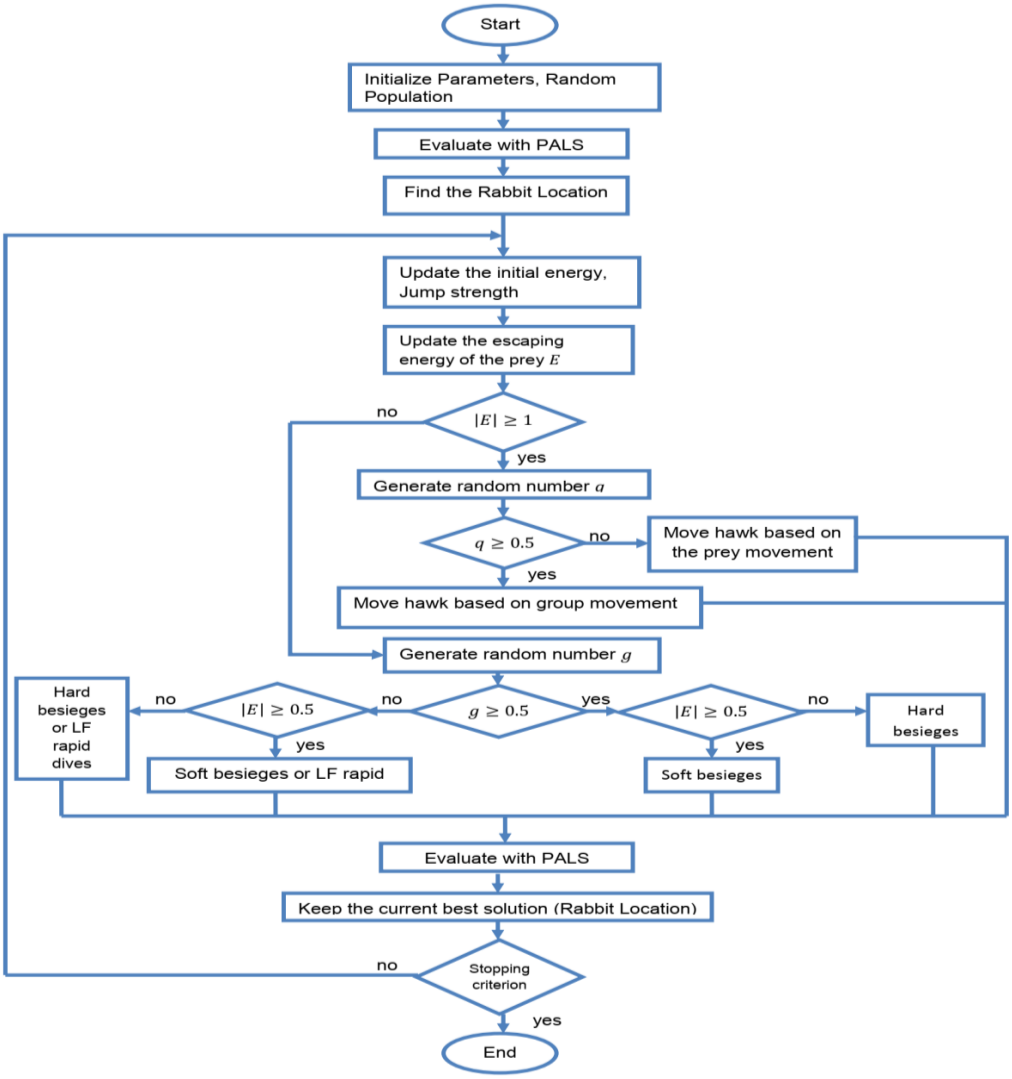

Fig.2 PHHO Flowchart.

5. Experiments and Validation

5.1. Instances of Benchmark

In this section, we validate the performance of the proposed hybrid algorithm. GenFrag [17] is selected for testing the efficiency of PHHO. All benchmark instances are downloaded from the National Center for Biotechnology Information (NCBI) [18]. The selected instances are characterized as:

- "X60189" set: clusters with a length of 3835 bases of fibronectin type III. These clusters are
- founded in the complex class III region of human histocompatibility.
- "M15421" set: clusters with a length 10,089 bases. These clusters are founded in the apolipoprotein B gene of humans.

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

For summarization, Table 1 shows the general characteristics of selected GenFrag instances. The columns of this table are the name of the instances, the coverage of each instance, the mean fragment length, the number of fragments, and the length of the original sequence.

Table 1. GenFrag Instances

Instances	Coverage	Mean Fragment Length	Number of Fragments	Original 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
M15421(5)	5	398	127	10,089
M15421(6)	6	350	173	10,089
M15421(7)	7	383	177	10,089

Table 2. The best results founded by PHHO and the compared algorithms.

Instance	Optimal	PHHO	**ABC** [9]	**GA [9]**	**GAG_{50} [10]**	**PALS [19]**	**FA [11]**
X60189(4)	11,478	11,478	11478	11478	11478	11204	11478
X60189(5)	14,161	14,161	14016	13502	13553	12898	14075
X60189(6)	18,301	18,301	18239	17688	17866	16992	18097
X60189(7)	21,271	21,271	21184	20884	20884	20424	20898
M15421(5)	38,746	38,746	38423	37714	37932	36540	37743
M15421(6)	48,052	47896	47515	46949	47152	45773	47033
M15421(7)	55,171	54870	54607	52695	52702	51454	51509

Doi :https://doi.org/10.54216/FPA.070201
Received: December 09, 2021 Accepted: April 3, 2022

5.2. Experimental Results

In the beginning, we examine the effect of PALS on the performance of the original HHO. As shown in Figure 3, the hybridization between PALS and HHO significantly affects the performance of the original HHO. Besides, PHHO is compared with popular algorithms from literatures, including the original Artificial Bee Colony (ABC) [9], Genetic Algorithm (GA) [9], GAG50 [10], PALS [19], and Firefly Algorithm (FA) [11]. The proposed algorithm is implemented in object-oriented Java in eclipse IDE. The number of iterations is set to 300 while the number of search agents is set to 30. The cutoff threshold is set to 30. The other algorithms' parameters are kept as originally suggested by the authors. All experiments are carried out on a 64-bit operating system with a 2.60 GHz CPU and 6 GB RAM. Table 2 shows the best solution founded by PHHO and the compared algorithms. The column of "Optimal" represents the optimal solution founded by LKH algorithm [20]. As observed, the proposed hybrid algorithm reaches optimal solutions in most cases. In particular, the proposed algorithm and the other comparators (except **PALS)** are able to reach the optimal solution for the " **X60189(4)"** instance. The proposed algorithm achieves the optimal solutions for the instances **"X60189(5)", "X60189(6)", "X60189(7)",** and **" M15421(5)".** Meanwhile, the other algorithms can't reach the optimal solution for these instances. For **"M15421(6)"** and **" M15421(7)"**, the proposed algorithm reaches the best solution comparing to the other compared algorithms.

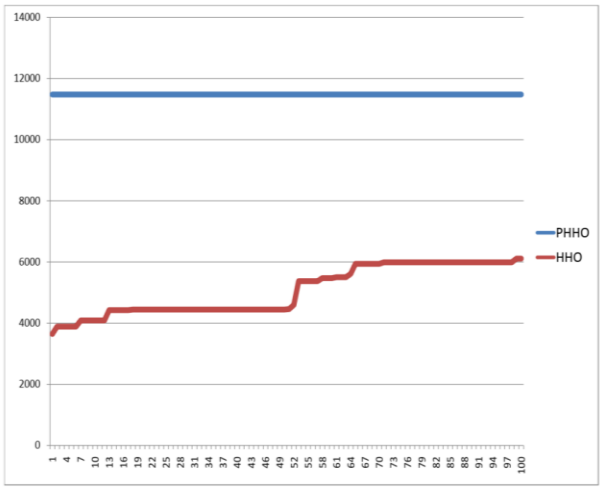

Fig. 3 HHO Vs. PHHO for "X60189(4)" instance.

6. Conclusion and Future Work

Coverage is a critical issue in military IoT applications. In this paper, MFO is used for detecting military intrusions with additional constraints to the number of sensors covering them. The grid-based area division is employed as it is more suitable for all types of surveillance sensors, not only the disc-shaped. MFO is compared

with other metaheuristic algorithms. The experiment results point out the efficiency and consistency of the proposed algorithm against others. The simulation indicates the ability and the fastness of MFO for reactive allocation of unmanned vehicles. For future work, we suggest several improvements can be made to the proposed algorithm, such as using Fuzzy Logic (FL) to determine the MFO parameters. Also, MFO can be applied to other types of military optimization problems like emergency response, sensor deployment, and cyber defense.

References

[1] Knetsch, C. W., van der Veer, E. M., Henkel, C., & Taschner, P. (2019). DNA Sequencing. In *Molecular Diagnostics* (pp. 339-360). Springer, Singapore.

[2] Huang, X., & Madan, A. (1999). CAP3: A DNA sequence assembly program. *Genome research*, *9*(9), 868-877.

[3] Green, P. (2009). Phrap, version 1.090518; 2009.

[4] Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., ... & Anson, E. L. (2000). A whole-genome assembly of Drosophila. *Science*, *287*(5461), 2196-2204.

[5] Sutton, G. G., White, O., Adams, M. D., & Kerlavage, A. R. (1995). TIGR Assembler: A new tool for assembling large shotgun sequencing projects. *Genome Science and Technology*, *1*(1), 9-19.

[6] Chen, T., & Skiena, S. S. (2000). A case study in genome-level fragment assembly. *Bioinformatics*, *16*(6), 494-500.

[7] Firoz, J. S., Rahman, M. S., & Saha, T. K. (2012, July). Bee algorithms for solving DNA fragment assembly problem with noisy and noiseless data. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation* ,201-208.

[8] Minetti, G., Alba, E., & Luque, G. (2008). Seeding strategies and recombination operators for solving the DNA fragment assembly problem. *Information Processing Letters*, *108*(3), 94-100.

[9] Ezzeddine, A. B., Kasala, S., & Navrat, P. (2014). Applying the firefly approach to the DNA fragments assembly problem. In *Annales Univ. Sci. Budapest., Sect. Comp*, 69-81.

[10] Allaoui, M., Ahiod, B., & El Yafrani, M. (2018). A hybrid crow search algorithm for solving the DNA fragment assembly problem. *Expert Systems with Applications*, *102*, 44-56.

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

[12] Earl, D., Bradnam, K., John, J. S., Darling, A., Lin, D., Fass, J., ... & Nguyen, N. (2011). Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome research*, *21*(12), 2224-2241.

[13] Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future generation computer systems*, *97*, 849-872.

[14] Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic algorithms: A comprehensive review. In *Computational intelligence for multimedia big data on the cloud with engineering applications* (pp. 185-231). Academic Press.

[15] Parsons, R. J., Forrest, S., & Burks, C. (1995). Genetic algorithms, operators, and DNA fragment assembly. *Machine Learning*, *21*(1-2), 11-33.

[16] Alba, E., & Luque, G. (2007, April). A new local search algorithm for the DNA fragment assembly problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization* (pp. 1-12). Springer, Berlin, Heidelberg.

[17] Engle, M. L., & Burks, C. (1994). GenFrag 2.1: new features for more robust fragment assembly benchmarks. *Bioinformatics*, *10*(5), 567-568.

[18] National Center for Biotechnology Information. (n.d.). Retrieved March 9, 2020, from https://www.ncbi.nlm.nih.gov/

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

[20] Mallén-Fullerton, G. M., Hughes, J. A., Houghten, S., & Fernández-Anaya, G. (2013). Benchmark datasets for the DNA fragment assembly problem. *International Journal of Bio-Inspired Computation*, *5*(6), 384-394.

Author Biography

Osama Maher completed his Bachelor's degree in the faculty of Engineering, Fayoum University, Egypt. He applied to the Master of Computer Science at Helwan University. Osama has a good background in Familiar incident response and bioinformatics. He uses the Internet of Things (IoT) and Industrial IoT systems and has a solid background in Linux/Unix and Windows systems.