IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 26, 2020, accepted December 11, 2020, date of publication December 14, 2020,
date of current version December 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3044857

An Efficient-Assembler Whale Optimization Algorithm for DNA Fragment Assembly Problem: Analysis and Validations

MOHAMED ABDEL-BASSET[1], (Senior Member, IEEE), REDA MOHAMED[1], KARAM M. SALLAM[1], RIPON K. CHAKRABORTTY[2], (Member, IEEE), AND MICHAEL J. RYAN[2], (Senior Member, IEEE)

[1]Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt
[2]Capability Systems Centre, School of Engineering and Information Technology, UNSW Canberra at the Australian Defence Force Academy, Campbell, ACT 2612, Australia

Corresponding author: Reda Mohamed (redamoh@zu.edu.eg)

ABSTRACT The study of deoxyribonucleic acid (DNA) is crucial in many fields, including medicine, biology, zoology, agriculture, and forensics. Since reading a DNA sequence is onerous because of its massive length, it is common in many DNA analysis applications to divide DNA strands into small segments or fragments which, after analysis, must be reassembled. Since this reassembly takes a non-specific polynomial time to solve, the DNA fragment assembly problem (DFAP) is NP-hard. This paper proposes a new assembler for tackling the DFAP based on the overlap-layout-consensus (OLC) approach. The proposed assembler adapts a discrete whale optimization algorithm (DWOA) using standard operators adopted from evolutionary algorithms to simulate the strategy adopted by humpback whales when searching for prey. For the first time, we formulate the behaviors of whales to be applied directly to any discrete optimization problem based on three primary operations: a swap-based best-position operator, an ordered crossover operator, and selection of a random whale operation to perform the exploitation and exploration phases of the algorithm. These operations were carefully designed to preserve the methodology of the original whale algorithm. DFAP is a multi-objective problem that seeks to reach the optimal order of segments that maximizes the overlap score and minimizes the number of contigs (set of overlapping DNA segments) to compose a one-contig DNA strand. Existing local search methods, such as problem aware local search (PALS) many non-conflicting movements (PALS2-many), suffer from being trapped in local optima. Hence, the integration of DWOA with PALS2-many improves the search capability for finding the optimal order of fragments. In addition, we propose a new variation of PALS2-many that achieves simultaneously the two objectives of DFAP. Our proposed DWOA was compared with a number of the most recent robust assemblers: a hybrid crow search algorithm for solving the DFAP (CSA-P2M*Fit), P2M*Fit, and a hybrid genetic algorithm (GA-P2M*Fit). The experimental results and statistical analyses of the proposed DWOA on thirty benchmark instances show that DWOA significantly outperforms those algorithms in reaching fewer contigs, in addition to being competitive with CSA-P2M*Fit and superior to P2M*Fit and GA-P2M*Fit for the overlap score.

INDEX TERMS DNA sequence, DNA fragments assembly problem, overlap-layout-consensus, whale optimization algorithm.

I. INTRODUCTION

Progress in the study of deoxyribonucleic acid (DNA) has allowed the early detection and prediction of an individual's exposure to many diseases including as cancer [1]–[3] and autoimmune [4] diseases. DNA analysis has been further extended to the fields of forensics and crime detection [5]–[8], genetic engineering and agriculture (i.e., improving the productivity of crops) [9]–[11]. Despite such wide-spread applications of DNA analysis, reading the complete DNA sequence is still an onerous task due to the massive length of DNA strands—human DNA is estimated to

The associate editor coordinating the review of this manuscript and approving it for publication was Shuai Liu.

contain about 3.2 billion nucleotides [12], [13]. So, a standard procedure is to divide the DNA strands into small segments or fragments at random positions to facilitate the reading process. Once the analysis is complete, the DNA fragments must be combined back into the original DNA sequence—the process is referred to as the DNA fragment assembly problem (DFAP). In DFAP, the main objective is to find the optimal order of the fragments to reassemble the original DNA sequence; the layout phase is therefore considered as the core of DFAP.

In order to ensure the accuracy of any reassembled DNA, the two main objectives of DFAP are to attain the optimal order of the fragments that combine to form the original DNA while maximizing the overlap score among these fragments and minimizing the number of contigs (set of overlapping DNA segments). The traditional approaches of DFAP try all the possible fragment combinations in order to detect the best combination. For F fragments, there are $2^F F!$ possible combinations such that solution time increases exponentially and it may take years to find the exact solution [12]. Due to the significant success of meta-heuristic algorithms in solving such problems in a reasonable time [14]–[21], this work was motivated to adapt those algorithms for solving DFAP in order to overcome the computational time and accuracy problems which plague traditional methods.

Recently, a meta-heuristic algorithm [22] known as whale optimization algorithm (WOA) has been proposed for tackling continuous optimization problems [23]–[25]. WOA has the advantages over a large number of the other meta-heuristic algorithms that motivate investigation of its use for tackling the DNA fragment assembly problem:

- Having two exploitation capabilities allows the whales to quickly move toward the optimal solution.
- Terminating its exploration capability after the first half of the iterations assists in accelerating convergence due to reducing the diversity between the members of the population; noting that this is considered a disadvantage for a problem with several local minima and a global minimum that is not easy to reach, as found in such problems as parameter extraction problem of double- and triple-diode models for solar photovoltaic systems [26].
- Easy to understand and implement. In addition, to the best of our knowledge, there is no research tackling this problem using WOA.

Recently, several variants of the WOA have been proposed for tackling various discrete optimization problems. In [27] A binary version of the WOA has been proposed for tackling the binary optimization problems, specifically three engineering optimization problems and a real-world travelling salesman problem. Mafarja [28] integrated the WOA with simulated annealing (SA) to address the feature selection problem; the SA was used to improve the quality of the best-so-far solution after a number of iterations until accelerating convergence toward the optimal solution. WOA was also integrated with the quantum theorem [29] to improve the diversification and intensification of the standard WOA for the feature selection problem. Further, WOA was improved by the Lévy flight strategy and the local search strategy (LSS) [30] for tackling the single and multidimensional 0-1 knapsack problems. WOA has also been suggested in [31] for tackling the clustering problem in data mining. Abdel-Basset [32] modified the WOA and hybridized this modified version with a LSS for addressing the scheduling of the multimedia data objects. Further, in [33] the green job scheduling problem was tackled by proposing a discrete version of the WOA. Jiang, *et al.* [34] improved the WOA for tackling the energy-efficient scheduling problem; the improvement was based on the dispatch rules, a nonlinear convergence factor, and a mutation operation (MO).

After reviewing the recent published variants of the WOA, we found that no variant has been proposed for addressing DFAP. Therefore, in this paper, we propose a new discrete variant based on adapting the behaviors of the WOA under relevant genetic operators for tackling DFAP.

At the outset, the performance of the standard WOA mapped using the largest position value (LPV) technique is initially proposed for addressing DFAP. LPV arranges the continuous positions of the whale in descending order. The largest position value is mapped to 1, the second-largest position value is mapped to 2, and so on. However, the performance of the WOA under this mapping technique is poor in comparison to some of the recent robust algorithms as shown in the results section. Recently, a new trend has appeared in several reported works, such as the crow search algorithm [35] and Faris *et al.* [36] to convert continuous optimization algorithms into discrete versions by borrowing some adequate genetic operators to simulate the nature of the standard algorithm for tackling combinatorial problems. This trend, in addition to the poor performance of the standard WOA, motivates us to propose a discrete version of the WOA by borrowing some relevant genetic operators for tackling the DFAP. Specifically, the discrete version of WOA (DWOA) is proposed under a number of genetic operators that are utilized to mimic the behavior of the WOA discretely:

- the swap-based best position operator to mimic the action of encircling prey;
- the ordered crossover operator to mimic the bubble-net attacking method; and
- random positions are used to search for prey.

The DWOA was compared with the standard version and a number of the well-known robust algorithms mapped using LPV and the results of the comparison show the efficacy of the DWOA when solving the DFAP. Moreover, the DFAP is solved under two objectives: minimizing the number of contigs with maximizing the overlap score. Therefore, at the first *subIter* iterations, DWOA is integrated with the PALS2-many technique [37] applied to search the best order of the fragments that minimizes the number of the contigs within this number of iterations. PALS2-many is summarized as follows:

1) The algorithm generates a neighborhood solution (NS) by a movement that reverses the sub-permutation between two different positions.
2) Next, the variations are calculated in the overlap score and the number of contigs between the current solution and the NS for only the swapping fragments. If the variation in the number of contigs is minimized or the overlap core is maximized with preserving the number of contigs, this movement is stored in a list.
3) Then, the algorithm selects many non-conflicting movements from the list that minimize the number of contigs and applies them on the current solution.
4) Finally, the previous three steps are continuously executed until there is no movement improving the current solution.

Then, within the remainder number of the iterations, an improvement on the PALS2-many was proposed and called PALS2-many-based fitness and contig (PMFC). The PMFC strategy applied to the current solution only the movements that increase the overlap while reducing or keeping the number of contigs. This new variant of DWOA that used a local search method to improve its quality in terms of the number of contigs and overlap score is abbreviated as DWOA-LS. The proposed algorithm (DWOA-LS) works on maximizing the overlap score based on the DWOA considering as the man objective that need to be optimized in our research as shown in most of the papers in the literature. While, within the first *subIter* iterations, the LS works on finding the order of the fragments that minimizes the number of contigs with an overlap score higher than obtained by the DWOA. Afterwards, within the rest of the iterations to maximize the overlap score considering the main objective of our research, the LS is adapted to search for the highest overlap score with preserving the number of contigs, or minimizing it. In general, the LS is employed at the first *subIter* iterations to find the order that will minimize the number of contigs with preserving the overlap score. For the remainder of the iterations, the LS will be functionalized to optimizing the overlap score with preserving the number of the contigs.

Generally, the main contributions and novelty of this work are:

1) Development of a discrete WOA (DWOA) for solving DFAP, that borrows some of the standard operators adopted from evolutionary algorithms to mimic the behaviors of humpback whales.
2) Incorporation of two advanced local search strategies (PALS2-many and an improved variant of PALS2-many) with DWOA (DWOA-LS) to boost the searching capability.
3) Conduct of a rigorous experimental analysis and comparison with the proposed DWOA against other existing assemblers. To do so, thirty instances were considered in terms of overlap score, the number of contigs, and a new evaluation function.
4) The investigation shows that DWOA outperforms all other algorithms used in the comparison, when considering two objectives together: the first objective is to minimize the number of contigs, and the second objective is to maximize the overlap score.

The remainder of this paper is organized as follows. Section II summarizes the related work of the DFAP. Section III presents the DNA fragment assembly problem. Section IV overviews the standard WOA. Section V illustrates the proposed approach to adapt WOA to solve the DFAP. Section VI presents the discussion and the experimental results of the proposed method for addressing DFAP on three sets of standard benchmarks. Section VII draws conclusions about the proposed approach and highlights some potential future work.

II. RELATED WORK

Alba and Luque [38] presented a heuristic algorithm called problem aware local search (PALS) that can obtain near-optimal solutions for DFAP better than the existing assemblers including PMA [39] and available commercial packages such as CAP3 [40]. Nonetheless, their proposed heuristic algorithm was still trapped in local optima and consequently converged slowly towards the optimal solution, particularly for large-scale DFAPs. Therefore, researchers have been encouraged to find new and effective methods for problems involving larger numbers of fragments. The emergence of metaheuristic algorithms and their promising success in tackling many optimization problems [41]–[47] has attracted many researchers to look to use such techniques for solving DFAP. One of the first algorithms that solved DFAP was a genetic algorithm (GA) proposed by Parsons *et al.* [48] using sorted-order and traditional permutation representations. Their results demonstrated that the edge-recombination crossover works better for such a problem, so they incorporated the permutation representation and the edge-recombination operation in a GA which produced better results than a greedy algorithm.

Nebro *et al.* [49] proposed a GA and solved the DFAP with the aid of a computing grid comprising 150 computers, reducing computing time from days to hours. Further, Hughes *et al.* [50] presented three variations of GA based on ring species, island model, and recentering-restarting to maximize the overlap score and obtain high-quality orderings. They integrated two heuristics, including 2-opt and Lin-Kernighan since one heuristic alone can become trapped in local optima. They concluded that the recentering-restarting variations works better with their proposed heuristic. However, in most of the solved instances, the algorithm does not attain the optimum overlap.

Bucur [51] designed an advanced GA by employing segmented permutations for representing candidate solutions. Their algorithm was verified using only three instances of DFAP of medium sizes. More recently, Rathee *et al.* [52] incorporated the quantum computing concept with GA (QGFA) to carry out DNA fragment assembly with an overlap-layout-consensus process. The performance of QGFA was assessed against several well-known algorithms, with the results showing that QGFA performs better that these algorithms for both the number of contigs and the overlap score obtained. In [53], three efficient algorithms (GA, simulated annealing (SA), and PALS) were proposed for handling noisy and noiseless DFAP instances. Among those meta-heuristics, SA demonstrated the best results for noiseless DFAP instances, while GA showed better performance for DFAP in the presence of noise.

Particle swarm optimization (PSO) is another meta-heuristic algorithm that is commonly used in solving DFAP. Some of the works done for tackling the DFAP based on PSO will be reviewed within this paragraph. Rajagopal and Sankareswaran [54] studied three variations of PSO, including constant and dynamic inertia weight, and adaptive PSO, with the adaptive PSO providing superior results. In [55], six variations of PSO were introduced using two seeding algorithms to generate the population and variable neighborhood search (VNS). The results showed that combining the SA, tabu search, and VNS with PSO is the best variant. Some authors have resorted to hybridization as a trend for tackling DFAP. Mallén-Fullerton and Fernandez-Anaya [56] combined PSO with the differential evolution (DE) algorithm. Further, Huang *et al.* [57] integrated PSO with the SA algorithm which was shown to outperform PSO, albeit with an increase in computational time.

Vidal and Olivera [58] presented a firefly algorithm (FA) [59] on a GPU (DFA-GPU) for DFAP. A local search (LS) is combined with DFA-GPU, which provides a parallel model for tackling different DFAP instances without degradation in the performance and time-consuming. For the suggested crow search algorithm (CSA) in [35], their fitness function only considered the overlap score, without considering the number of contigs which, as a consequence, results in inferior performance. In addition, Ali [60] proposed a discrete particle swarm optimization (DPSO) based on a new updating rules known as probabilistic edge recombination (PER) for tackling the layout stage in the OLC DFAP. PER operator creates a new permutation by considering relative ordering of DNA fragments. In addition, Ali, within the same research, created another variant of DPSO combined with PALS to improve the exploitation capability for reaching better outcomes; this variant was called quick-PALS.

Furthermore, the memetic gravitational search algorithm (MGSA) [61] is proposed for tackling the DFAP based on the OLC approach and used the tabu search to initialize the population. Moreover, MGSA used time-varying maximum velocities to increase the diversity among the members of the population to reduce the probability of stuck into local minima problem. Finally, MGSA was integrated with the SA-based variable neighborhood search to improve the accuracy of the best solution obtained by MGSA. The MGSA was validated on 19 DFAP instances in an attempt to maximize the overlap score among the fragments of each sequence. However, the MGSA doesn't take in consideration the number of contigs, which is its main limitation.

Ülker [62] adapted the harmony search (HS) algorithm for DFAP. Because HS was proposed to address the continuous optimization problem and the DFAP is a combinatorial one, the smallest position value was used to convert the continuous solutions generated by HS into permutation ones to be adequate for tackling this combinatorial problem: DFAP. The performance of HS was observed using three real DNA datasets. Indumathy [63] adapted the cuckoo search (CS) algorithm to reconstruct the original sequence of the segmented DNA as the first attempt to apply this algorithm on the DFAP. The CS algorithmwas observed using nine instances and compared with a number of variants of the PSO. The experimental results show the superiority of the CS over those variants. Other metaheuristics developed for DFAP include the ant colony system algorithm [64], and the bee algorithm [65].

All the algorithms mentioned in the literature dealt with DFAP by improving one of the following two objectives: (1) minimizing the number of contigs only, or (2) maximizing the overlap score only. This is at odds with the nature of the problem that needs to achieve the two objectives simultaneously when looking for the best order of the DNA fragments while maximizing the overlap score among the fragments in this order. Specifically, the major problems that affect the algorithms developed in the literature are summarized as follows:

- Most of approaches have difficulty in avoiding becoming trapped in local optima.
- The efficacy of most approaches has not been tested on a sufficient number of large-scale instances.
- Existing algorithms can obtain the optimal overlap for several cases, but the number of subsequent contigs is too large.

The significance of quick, reliable DNA analysis and the shortcomings of existing approaches motivate us to suggest an efficient assembler based on DWOA for solving the DFAP. The efficient assembler DWOA-LS presented here works on finding the best order of the fragments while maximizing the overlap among them.

III. DNA FRAGMENT ASSEMBLY PROBLEM

Deoxyribonucleic acid (DNA) is the hereditary material that stores the information required to create all living organisms. DNA consists of four chemical bases, including Adenine (A), Cytosine (C), Thymine (T), and Guanine (G). The sequence of the four letters (A, G, C, and T) indicates the available information for building the living organism. The letters join in pairs, A with T, C with G, to create base pairs. Each pair is tied to a sugar molecule and a phosphate molecule to compose a nucleotide. A nucleotide resembles a ladder comprising of two long strands that make a spiral called a double helix [35], [66].

In the field of computational biology, these sequences are used to extract the function of information coded in DNA. The human genome consists of a vast number of bases (about 3.2 billion), and the current sequencing technologies cannot read more than 1000 bases. Accordingly, a shotgun sequencing strategy has been used to overcome that problem by breaking the long DNA sequence into smaller pieces called fragments or segments. These fragments are sequenced randomly by machine so that the original order and orientation of them is lost. The fragments have to be reassembled based on the overlap among them to regain the original order of the DNA. The assembler has to calculate the overlap between all possible pairs of fragments to reassemble the ones that have the highest similarity score to compose a contig, which consists of a set of contiguous and overlapping fragments. The problem is known as the DNA fragment assembly problem (DFAP) and the challenge is to reassemble the contigs successfully to retrieve the original DNA.

The traditional assembly approach involves three stages: overlap, layout, and consensus, which is known as the overlap-layout-consensus (OLC) approach. In the overlap step, the assembler calculates the overlap score between all possible fragments. Detecting the highest overlap score between the prefix of one fragment and the suffix of another is the objective of this step. The semi-global alignment method is adopted and is implemented using dynamic programming [3]–[5]. In the layout step, the order of the fragments that maximizes the sum of all the overlaps of each adjusted fragment is reassembled until the original DNA sequence is obtained. Finally, in the consensus step, the order of the fragments from the layout step is used to form the complete DNA. These phases are illustrated in Fig. 1.

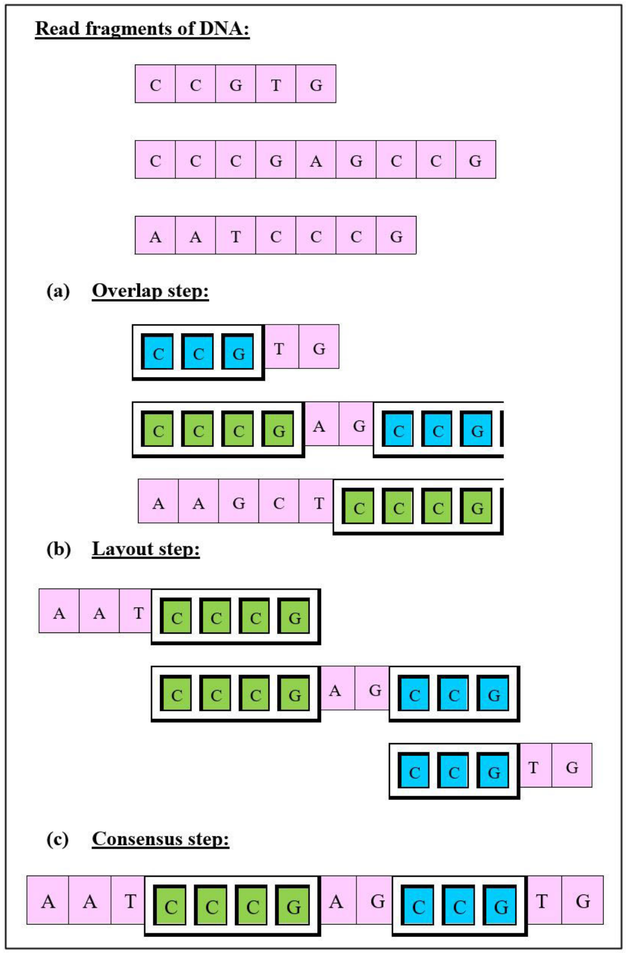


FIGURE 1. Illustration of the OLC approach.

IV. STANDARD WHALE OPTIMIZATION ALGORITHM: OVERVIEW

In WOA, Mirjalili and Lewis [22] mimicked the actions and behaviors of humpback whales, which use an astounding feeding method called the bubble-net approach when attacking their victim or prey. They surround the victim in a spiral shape and then swim up to the surface in a shrinking circle. WOA mimics this hunting tradeoff between a spiral model and a shrinking encircling prey with a probability of 50% to update the position of the whale. The mathematical model for the encircling mechanism is as follows:

$$\vec{Z}(t+1) = \vec{Z^*}(t) - \vec{A} * \vec{D} \tag{1}$$

$$\vec{A} = 2 * a * rand - a \tag{2}$$

$$a = 2 - 2 * \frac{t}{t_{maxIter}} \tag{3}$$

$$\overrightarrow{Dist} = |\vec{C} * \vec{Z^*}(t) - \vec{Z}(t)| \tag{4}$$

$$\vec{C} = 2 * rand \tag{5}$$

where $\vec{Z}$ is the position vector of the current whale, t the current iteration, $\vec{Z^*}$ the position vector of the best whale in the population, $rand$ a random number between [0, 1], $t_{maxIter}$ to maximum number of iterations, and a a distance control parameter linearly decreased from 2 to 0 [22]. The spiral model tries to mimic the helix-shaped movement of whales. The mathematical model of a spiral shape is as follows:

$$\vec{Z}(t+1) = \vec{Z^*}(t) + cos(2 * \pi * l) * e^{l*b} * \overrightarrow{Dist} \tag{6}$$

$$\overrightarrow{Dist} = |\vec{Z^*}(t) - \vec{Z}(t)| \tag{7}$$

where $\overrightarrow{Dist}$ is a vector used to store the absolute distance between $\vec{Z^*}(t)$ and $\vec{Z}(t)$, l a numerical value created randomly between [-1, 1] and b a fixed value to depict the logarithmic spiral shape. To search for the prey in another direction of the search space, WOA uses a random whale from the population to update the position of the current whale in the exploration phase. If $\vec{A}$ is greater than 1, then the current whale is updated according to a random whale from the population. The mathematical model of the search for the prey is as follows:

$$\vec{Z}(t+1) = \vec{Z^*}(t) - \vec{A} * \vec{D} \tag{8}$$

$$\overrightarrow{Dist} = |\vec{C} * \vec{Z^*}_{ind} - \vec{Z}(t)| \tag{9}$$

where $\vec{Z}_{ind}$ is a vector including the position of a whale with an index of ind in the current population; this index is randomly selected between 1 and N to enable the WOA to explore other regions within the search space to avoid becoming trapped into local minima. With the significant success achieved by WOA when solving many optimization problems, in this research, it is adapted for the first time in this work to address the DFAP as discussed later.

V. THE PROPOSED APPROACH: DWOA-LS ALGORITHM

This section provides a full explanation and illustration of the proposed DWOA-LS algorithm. The fundamental components of this proposed solution approach are: initialization, fitness evaluation, the DWOA for DFAP, and PALS as a local search approach. Each of these components is described in the following subsections.

A. INITIALIZATION

DFAP is a combinatorial problem, seeking to increase the overlap among the adjacent fragments, while the ultimate goal is to produce a one-contig DNA. To provide an effective solution representation, a proper understanding of this problem is essential. The set of fragments is enumerated from 1 to N, where N is the maximum number of fragments. Then, a possible solution to DFAP is to rearrange the set of numbers from 1 to N. The identification of the optimal order of the fragments requires an examination of the permutations of the numbers assigned to those fragments. Now, to solve this DFAP by using the proposed DWOA-LS we assume that each whale carries a solution to the problem. We randomly initialize the population of M whales, while each whale is described by a position vector of size N containing a permutation of numbers assigned to the fragments, which represent a solution to DFAP. Each position in the whale position vector should have a different fragment number.

B. OBJECTIVE FUNCTION

An objective function plays an essential role in DWOA-LS to reach the best solution for a given problem. In this case, the objective function calculates the fitness value of each whale in the population. The whale that has the best fitness value is identified as the best solution. In DFAP, the main objective is getting the optimal arrangement of the fragments, which achieves the highest overlap score and reduces the number of contigs. In the proposed DWOA, the evaluation is based on the overlap score only, and the minimization of the number of contigs is considered as an objective for the two versions of the PALS2-many method. Therefore, to calculate the overlap score of each whale estimated by the DWOA to find the nearest one to the optimal solution:

$$F(\overrightarrow{Z}(t)) = \sum_{d=0}^{N-2} w(f_d, f_{d+1}) \tag{10}$$

where $F(\overrightarrow{Z}(t))$ represents the fitness values of the current whale $\overrightarrow{Z}(t)$, and $w(f_d, f_{d+1})$ the overlap score between any two consecutive fragments. In each possible order estimated by an algorithm, this equation is used to estimate its quality by summing the count of the similar consecutive letters at the end of the first fragment f_d with the letters at the beginning of the second one $f_(d + 1)$, and the order with the highest overlap score is considered the best. For example, Fig. 1 includes three fragments, which need to be arranged to return the original DNA sequence. Therefore, the sum of the count of the similar letters between each two consecutive fragments is calculated, and the order that fulfills the highest score is considered the best as depicted in this figure. As identified previously, the overlap score has been calculated using semi-global alignment, implemented by a dynamic programming approach.

C. THE DISCRETE WHALE OPTIMIZATION ALGORITHM (DWOA)

The standard WOA was designed to address the continuous-search space problems and cannot, therefore, be used with the discrete-search space of the DFAP. In the literature, authors suggest converting continuous optimization-based algorithms to discrete ones by using mapping methods, such as the smallest position value (SPV) or the LPV. SPV arranges the continuous positions of the whale in ascending order so that the smallest position value is mapped to 1; the second smallest position value is mapped to 2, and so on. Unlike SPV, LPV arranges the continuous positions of the whale in descending order. The largest position value is mapped to 1, the second-largest position value is mapped to 2, and so on. From the experimental results presented here, the mapping of continuous search space into a discrete space isn't an effective way to solve DFAP. The main disadvantages of using the continuous algorithms mapped using LPV and SPV to solve the combinatorial problems are as follows:

- In combinatorial problems, generally, updating all the positions of the vector together may deteriorate the quality of the solution because it may need a small change to reach the optimal.
- SPV arranges the continuous values and the smallest one is given a value of 1, the second smallest a value of 2, and so on. However, if there are two continuous values are equal, then one will take a random value and subsequently this will convert the algorithm to a random search if a significant number of the continuous values are equal.

Therefore, a new trend is to redesign the standard algorithm to deal with combinatorial problems. Examples include: in [35], the crow search algorithm was redesigned with the ordered crossover operation, which was more suitable for DFAP; and Faris *et al.* [36] converted the addition operation in the salp swarm algorithm into a crossover operation providing improved results compared to selected standard algorithms. An essential stage in DWOA is the adaptation of Eqs. (1), (6), and (8) to be able to address the solution of discrete problems. In this work, the WOA is adapted using the following operators:

- The swap-based best position operator mimics the action of encircling prey.
- The ordered crossover operator mimics the bubble-net attacking methods.

- random positions are used to search for prey.

The adaptation of the previous three operations to simulate the actions performed by the whales is illustrated in detail in the next subsections.

D. ENCIRCLING PREY (EXPLOITATION PHASE)

In this phase, the whales encircle the prey as in Eq. (1). However, Eq. (1) is used for continuous problems only. Therefore, the swap-based best-position operator is used to imitate this action for the DFAP that is, to enable the current whale to update its position towards the best position or the optimum prey $\overrightarrow{Z}^*(t)$. To update the position of the whale $\overrightarrow{Z}(t+1)$, the following steps are executed:

- selecting a random position i from the best whale $\overrightarrow{Z}^*(t)$ and return the value v_i found in that position;
- searching for the value v_i in the current whale $\overrightarrow{Z}(t)$ and return its position j; and
- swapping the values in the positions i and j in the current whale $\overrightarrow{Z}(t)$.

This operator enables the current whale to move toward the victims gradually before attacking them, as illustrated in Fig. 2.

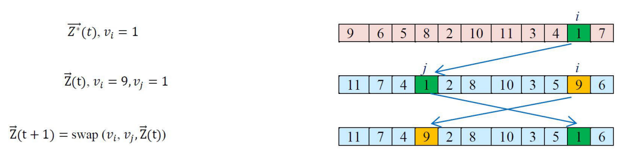


FIGURE 2. An illustrative example of a swap-based best-position operator between the prey and current whale.

If the value v_i in the best whale $\overrightarrow{Z}^*(t)$ is equal to the value v_i in the current whale $\overrightarrow{Z}(t)$, we select a random position j from the current whale $\overrightarrow{Z}(t)$. Then, we swap the two values in the positions i and j in the current whale $\overrightarrow{Z}(t)$ as shown in Fig. 3.

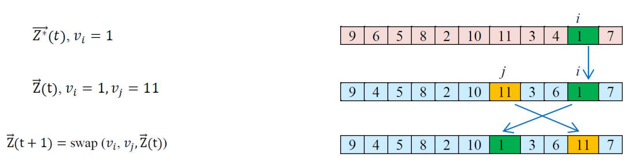


FIGURE 3. An illustrative example of a swap-based best position operator between the prey and current whale for equal values. The values in the position i in the current and best whale are equal, so another position j is selected from the current whale and swapped with ***i***.

E. SPIRAL BUBBLE-NET FEEDING METHOD (EXPLOITATION PHASE)

In this phase, the whales swim up towards the prey in a spiral shape as in Eq. (6). But to fit this for discrete problem, we have used an ordered crossover operator to simulate this action for DFAP. The ordered crossover operator is applied to the current whale $\overrightarrow{Z}(t)$ to modify its positions towards the best whale $\overrightarrow{Z}^*(t)$ found so far. To update the position of the whale $\overrightarrow{Z}(t+1)$, we follow these steps:

- Generate two random positions i and j within the current whale $\overrightarrow{Z}(t)$, such that $i < j$.
- Copy all the values between the two positions i and j from the best whale $\overrightarrow{Z}^*(t)$ to $\overrightarrow{Z}(t+1)$.
- Remove the previous values copied to $\overrightarrow{Z}(t+1)$ from the current whale and copy the remaining values in the order they appear from $\overrightarrow{Z}(t)$ to $\overrightarrow{Z}(t+1)$ to fill the positions before i then the positions after j.

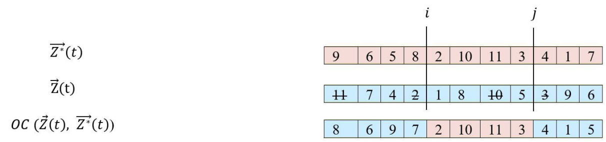


FIGURE 4. An illustrative example of the ordered crossover operator between the prey and current whale.

Fig. 4 describes the ordered crossover (OC) operation between the best whale and the current one. We replace the original equation (Eq. (6)) with the following equation 11:

$$\overrightarrow{Z}(t+1) = OC(\overrightarrow{Z}(t), \overrightarrow{Z}^*(t)) \tag{11}$$

F. SEARCH FOR PREY (EXPLORATION PHASE)

In the standard WOA, the current whale $\overrightarrow{Z}(t)$ moves towards a random whale selected from the population to search for the prey. However, if we apply this procedure for DFAP, the variation in population will be reduced. Therefore, to increase the diversity of the population, the current whale $\overrightarrow{Z}(t)$ is updated and randomly generated from the solution area of the problem to explore more promising solutions that were not discovered before. Based on the hunting behavior of the WOA, there is a probability of 50% of selecting between a spiral model and a shrinking encircling prey to update the position of the whale.

G. INTEGRATION OF DWOA WITH LOCAL SEARCH

In this section, we explain how to integrate DWOA with a heuristic method to boost its performance and improve the quality of the solutions. We used two variations of the PALS [38]. At first, we review the previous versions of PALS:

- The original PALS iteratively ameliorated a random solution by producing its neighborhood solutions. The neighborhood solution (NS) is generated by a movement that reverses the sub-permutation between two positions in the solution. Then, the algorithm calculates the variation in the overlap ($\triangle p$) and the number of contigs ($\triangle NC$), between the current solution and the NS for only the affected fragments. PALS tries all the possible movements and stores them in a list, and then selects a single movement that reduces or maintains the number of contigs while not decreasing the overlap. When several movements have the same minimum $\triangle NC$, PALS chooses the NS with the maximum $\triangle p$.

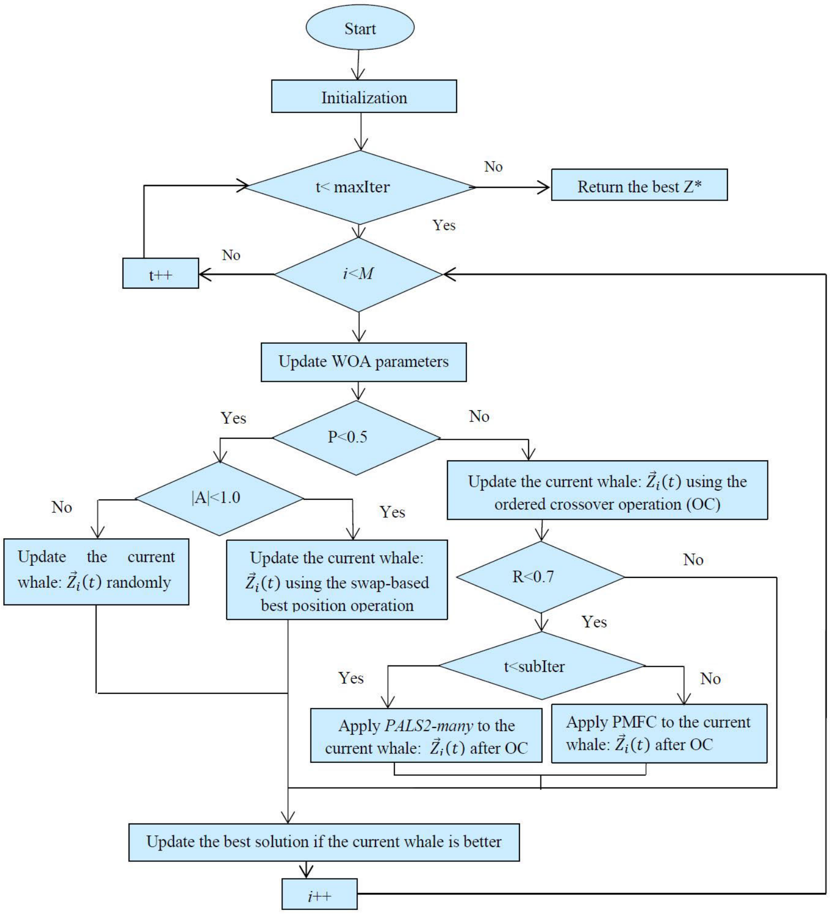


FIGURE 5. Flowchart of the proposed algorithm DWOA-LS for solving DFAP.

The drawbacks of PALS is that a particular NS may appear again through iterations and the calculations may be redone, which is time-consuming.

- In PALS2-many [37], the algorithm selects the movement that reduces $\triangle NC$, but in the case of having several movements with the same minimum $\triangle NC$, PALS2-many selects the NS with the lowest $\triangle p$. To speed up the algorithm, many non-conflicting movements are selected from the list, so that the algorithm reduces the number of calculations required. PALS2-many can produce a sub-optimal solution with minimum $\triangle NC$, but it can't reach the optimal overlap in large instances.
- PALS2M*Fit [35] is concerned with the movements that increase $\triangle p$. It can obtain the optimal overlap, but the number of contigs is large, especially for large instances, which conflicts with the ultimate objectives of achieving a single-contig DNA with optimal overlap.

The disadvantages of PALS2-Many are summarized as follows:

1) adding the movements that minimize the number of contigs to the list even if they minimize the overlap score; and
2) adding to the list the movements that preserve the number of contigs with maximizing the overlap score.

It should be noted that there is a case that isn't taken into consideration by PALS2-Many and PALS2M*Fit, in which the movements that will only maximize the overlap score by preserving or reducing the number of contigs. If those movements will minimize the number of contigs only, they must be discarded because one objective will be achieved, but not the other. Therefore, to tackle this issue in this work, an improvement is proposed on the PALS2-many called PALS2-many-based fitness and contig (PMFC). The PMFC strategy is applied to list the movements that increase the overlap $\triangle p$ while reducing or keeping $\triangle NC$. This improvement seeks to find the order that not only maximizes the overlap score among the fragments but also minimizes, or at least preserves, the number of contigs among them. This will help in reaching the near-optimal order of the fragments that reconstruct correctly the original DNA sequence.

The PALS approach tries to achieve the best contig and the PALS2M*Fit tries to attain the optimal overlap, neither of them attempt to achieve both objectives together. In this work, DWOA is incorporated with PALS2-many to exploit the search capability of DWOA and the improvement performed by PALS2-many to obtain better solutions. For a predetermined number of iterations (*subIter*), PALS2-many is applied to the new candidate whale $\overrightarrow{Z}(t+1)$ after the crossover operation with an objective of minimizing the number of contigs in the hope of finding one contig.For the remaining iterations, the PMFC strategy is applied to maximize the overlap score by preserving or minimizing the number of contigs. The switch between PALS2-many and PMFC enables DWOA of reaching the optimal order of fragments. Both PALS2-many and PMFC is performed with a local search probability smaller than R, where R is a random value in the range [0, 1]. Fig. 5 shows the flowchart of the proposed DWOA integrated with PALS2-many and PMFC as local search methods (DWOA-LS).

Algorithm 1 The Proposed DWOA-LS

1: Initialize the population of whales $Z_i, (i = 1, 2, 3, \ldots, n)$;
2: Evaluate the fitness of each whale;
3: Find the best whale Z^*;
4: $t = 0$;
5: **while** $t \leq t_{maxIter}$ **do**
6: **for** $i = 1 : n$ **do**
7: Update WOA parameters
8: **if** $p < 0.5$ **then**
9: **if** $|A| < 1$ **then**
10: Update $\overrightarrow{Z}(t+1)$ using swap best position operation;
11: **else**
12: Update $\overrightarrow{Z}(t+1)$ using random position;
13: **end if**
14: **else**
15: Update $\overrightarrow{Z}(t+1)$ using the modified ordered crossover operation;
16: Generate $R \in [0, 1]$;
17: **if** $R < LSP$ **then**
18: **if** $t < subIter$ **then**
Apply PALS2-many to $\overrightarrow{Z}(t+1)$;
19: **else**
20: Apply PMFC to $\overrightarrow{Z}(t+1)$;
21: **end if**
22: **end if**
23: **end if**
24: **end for**
25: Check the feasibility of the whale $\overrightarrow{Z}(t+1)$;
26: Update $\overrightarrow{Z}(t+1)$ in the population, if better;
27: Update the best whale $\overrightarrow{Z^*}$ with $\overrightarrow{Z}(t+1)$ if better;
28: $t \leftarrow t + 1$;
29: **end while**
30: **return** the best whale $\overrightarrow{Z^*}$

Algorithm 1 illustrates the steps of solving DFAP using DWOA-LS improved with the PALS2-many and PMFC. The first step, the population is initialized randomly. Then, the fitness value for each whale inside the population is calculated, and the whale that has the highest fitness value is indicated as the best whale and stored in $\overrightarrow{Z^*}$. In the next step, from line 4 to line 25 the whales update their positions using the swap-based best position operation, the ordered crossover operation, and the random whale through a number of iterations. In line 15 to line 23, the current whale is updated using the ordered crossover operation. Then PALS2-many and PMFC are applied to the current whale after the crossover operation with a local search probability (LSP) discussed in the parameter settings section. PALS2-many is applied for a specified number of iterations and PMFC is applied for the remaining iterations. The current whale is updated in the population if better. The best whale is updated through iterations. The algorithm satisfies a number of iterations. On completion, the algorithm returns the best obtained solution.

More illustration, in DWOA-LS, the DWOA strives to optimize the overlap score regarding the main objective that needs to be optimized for solving the DFAP as used in most of the papers in the literature, while the LS (PALS2-many and PMFC) is employed to optimize the number of contigs. In brief, DWOA will work to optimize the overlap score as the main objective, while LS strives to minimize the number of contigs with an overlap score higher than obtained by the DWOA within the first slice of the iteration that is smaller than *subIter*. While, within the rest of the iterations, LS based on PMFC seeks to maximize the overlap score by preserving, or reducing the number of contigs. Therefore, the DWOA will work on maximizing the overlap score, while the LS work also on maximizing the overlaps score with a constraint to ignore the solutions that will increase the number of contigs although the overlap score is maximized.

H. COMPUTATIONAL COMPLEXITY

The time complexity of DWOA-LS is observed in this section. Specifically, the time complexity of the proposed approach is based on the following:

1) Updating the positions in each generation that is based on:
   a) The number of whales, M.
   b) The number of fragments, N.
   c) Cost of the objective function, C_{obj}.
2) The number of the iterations $t_{maxIter}$.
3) Applying the local search strategy: PALS.

The time complexity of updating the current whales in big-O is of $O(t_{maxIter}NM)$. While the big-O of evaluating the whales is of $O(t_{maxIter}MC_{obj})$. Generally, the time complexity of DWOA is expressed as follows:

$$O(DWOA) = \begin{cases} O(t_{maxIter}NM) & \text{if } cost(N) > cost(C_{obj}) \\ O(t_{maxIter}MC_{obj} & \text{if } cost(N) < cost(C_{obj}) \end{cases} \tag{12}$$

In Eq. 12, the time complexity of the proposed algorithm relies on the cost of the objective function and the number of fragments. The time complexity of the local search strategy is about $O(N^2)$ for one iteration as described in the pseudo-code of the PALS2-many. Since PALS2-many is applied with a probability with our proposed approach, the number of times where this method is executed is not known. Therefore, in the worst case, assuming that this method is applied in all iterations, the time complexity of the proposed is estimated as follows:

$$\begin{aligned} O(DWOA-LS) &= O(DWOA) + O(PALS) \\ &= O(DWOA) + O(t_{maxIter}N^2M) \\ &= O(t_{maxIter}N^2M) \end{aligned} \tag{13}$$

Since the PALS has a higher growth rate in terms of time complexity of, the time complexity of the proposed algorithm in the worst case is $O(t_{maxIter}N^2\ M)$, which is quite significant. Therefore, time complexity is one of the main limitations of our proposed approach that needs to be improved in future work.

VI. EXPERIMENTS AND DISCUSSION

Several experiments have been conducted to assess the efficacy of our proposed DWOA-LS algorithm. Thirty benchmark instances are chosen for testing the DWOA-LS effectiveness. We perform all the experiments on a device equipped with Windows 7 ultimate platform with a 64-bit operating system, Intel® Core i3-2330M CPU @ 2.20 GHz, and 1 GB of RAM. DWOA-LS is implemented using the Java programming language. Statistical analyses are also introduced to validate the results. This experimental section is designed as follows. Subsection VI-A describes the DFAP benchmark instances used in the experiments. Subsection VI-B describes the parameter setting of DWOA-LS. Section VI-C evaluates the performance of the proposed DWOA-LS. Section VI-D compares DWOA-LS with the best three recent assemblers (based on our knowledge) suggested for solving DFAP. Section VI-E compares the proposed DWOA-LS with some others assemblers. Finally, section IV-F summarizes the conclusion of our experiments.

A. DESCRIPTION OF THE BENCHMARK INSTANCES

We examine the performance of DWOA-LS on three benchmark collections taken from [67]: GenFrag consisting of ten instances; DNAgen containing six instances, and f-series containing fourteen instances. Table 1 presents a description of the thirty instances in terms of coverage, average fragment length (AFL), number of fragments (NF), and the original sequence length (OSL). Here, the coverage is the summation of the bases found in all fragments divided by the total length of the original DNA sequence [35], which can be calculated by using Equation 14.

$$Coverage = \frac{\sum_{j=1}^{NF} \text{length of fragment} j}{\text{length of target fragment}} \tag{14}$$

The coverage value has to be greater than 1 to ensure that there is an overlap between the fragments to be used in the reassembling process. AFL ranges from 182 to 1003 bases; NF ranges from 25 to 1577 fragments. For simplicity, we provide an abbreviation for each instance which is used in the remainder of the paper.

TABLE 1. Description DFAP instances.

ID	Instances	Abbreviation	Coverage	AFL	NF	OSL
1	X60189(4)	X4	4	395	39	3835
2	X60189(5)	X5	5	286	48	3835
3	X60189(6)	X6	6	286	48	3835
4	X60189(7)	X7	7	387	68	3835
5	M15421(5)	M5	5	398	127	10089
6	M15421(6)	M6	6	350	173	10089
7	M15421(7)	M7	7	383	177	10089
8	J02459(7)	J7	7	405	352	20000
9	BX842596(4)	BX4	4	708	442	77292
10	BX842596(4)	BX7	7	703	773	77292
11	Acin1	AC1	26	182	307	2170
12	Acin2	AC2	3	1002	451	147200
13	Acin3	AC3	3	1001	601	200741
14	Acin5	AC5	2	1003	751	329958
15	Acin7	AC7	2	1003	901	426840
16	Acin9	AC9	7	1003	1049	156305
17	F25(305)	F305	-	307	25	7630
18	F25(400)	F400	-	400	25	10006
19	F25(500)	F500	-	500	27	13051
20	F50(315)	F315	-	315	50	15791
21	F50(412)	F412	-	412	50	20628
22	F50(498)	F498	-	498	50	24956
23	F100(307)	F307	-	307	100	30443
24	F100(415)	F415	-	415	100	-
25	F100(512)	F512	-	512	100	-
26	F508(354)	F508	-	354	508	-
27	F635(350)	F635	-	350	635	-
28	F737(355)	F737	-	355	737	-
29	F1343(354)	F1343	-	354	1343	-
30	F1577(354)	F1577	-	354	1577	-

B. PARAMETER SETTINGS

Parameter setting may affect the performance of the algorithm. So, several experiments were performed to detect the best values for the parameters. Six instances: M15421(5), M15421(6), M15421(7), J02459(7), BX4, and BX7 are used for tuning the population size (M), $subIter$, and LSP, with their results are introduced in Tables 2, 3 and 4, respectively. Considering the population size, different values are considered such as 5, 10, 20, and 30 on different benchmark instances. Table 2 shows that the population size 30 is better because using this population value enables the proposed algorithm to reach the optimal value in fewer iterations for six instances. The population size of 5 is the worst.

Considering $subIter$, to test the efficiency of the proposed algorithm, several experiments are conducted by considering $subIter = 20, 50, 70, 100, 120$ and 500, with the results presented in Table 3. Regarding $subIter$, at the outset, a value of 20 was selected randomly. With a cutoff value between the compared fragments equal to 50, the number of contigs was 8 for BX4 and 2 for the BX7, while the fitness values were 227682 and 444839 for those two instances, respectively. For a value of 50, the number of contigs didn't change, but the fitness values for BX4 and BX7 become 227878 and 445039, respectively. Because changes in the overlap scores were quite significant, another value of 70 was selected and changes in the overlap score were observed. Consequently, three other values of 100, 120, and 500 were selected and

TABLE 2. Tuning of the population size (Pop Size) parameter.

Datasets	Pop Size=5			Pop Size=10			Pop Size=20			Pop Size=30		
	Fit	NC	bestIter	Fit	NC	bestIter	Fit	NC	bestIter	Fit	NC	bestIter
M15421(5)	38746	1	280	38746	1	138	38746	1	85	38746	1	70
M15421(6)	48052	1	92	48052	1	80	48052	1	48	48052	1	41
M15421(7)	55171	1	325	55171	1	270	55171	1	190	55171	1	105
J02459(7)	116700	1	338	116700	1	253	116700	1	185	116700	1	109
BX4	227920	1	1220	227920	1	961	227920	1	450	227920	1	90
BX7	445422	2	1410	445422	2	1000	445422	2	800	445422	2	671

TABLE 3. Tuning of the ***subIter*** parameter.

Datasets	subIter=20		subIter=50		subIter=70		subIter=100		subIter=120		subIter=500	
	Fit	NC	Fit	NC	Fit	NC	Fit	NC	Fit	NC	Fit	NC
M15421(5)	38707	3	38717	3	38735	3	38741	3	38741	3	38746	3
M15421(6)	48052	2	48052	2	48052	2	48052	2	48052	2	48052	2
M15421(7)	55130	2	55156	2	55166	2	55170	2	55170	2	55170	3
J02459(7)	116544	1	116577	1	116540	1	116560	1	116565	1	116547	1
BX4	227682	8	227823	8	227878	8	227912	8	227912	8	227920	8
BX7	444839	2	445039	2	445183	2	445251	2	445293	2	445330	2

TABLE 4. Tuning of the ***LSP*** parameter.

Datasets	LSP=0.2			LSP=0.3			LSP=0.5			LSP=0.7			LSP=1.0		
	Fit	NC	Best Iter	Fit	NC	Best Iter	Fit	NC	Best Iter	Fit	NC	Best Iter	Fit	NC	Best Iter
M15421(5)	38746	1	169	38746	1	136	38746	1	112	38746	1	108	38746	1	108
M15421(6)	48052	1	24	48052	1	19	48052	1	10	48052	1	8	48052	1	6
M15421(7)	55171	1	270	55171	1	236	55171	1	182	55171	1	120	55171	1	112
J02459(7)	116700	1	169	116700	1	162	116700	1	115	116700	1	105	116700	1	105
BX4	227920	1	428	227920	1	323	227920	1	263	227920	1	161	227920	1	141
BX7	445422	2	2000	445422	2	1077	445422	2	550	445422	2	325	445422	2	280

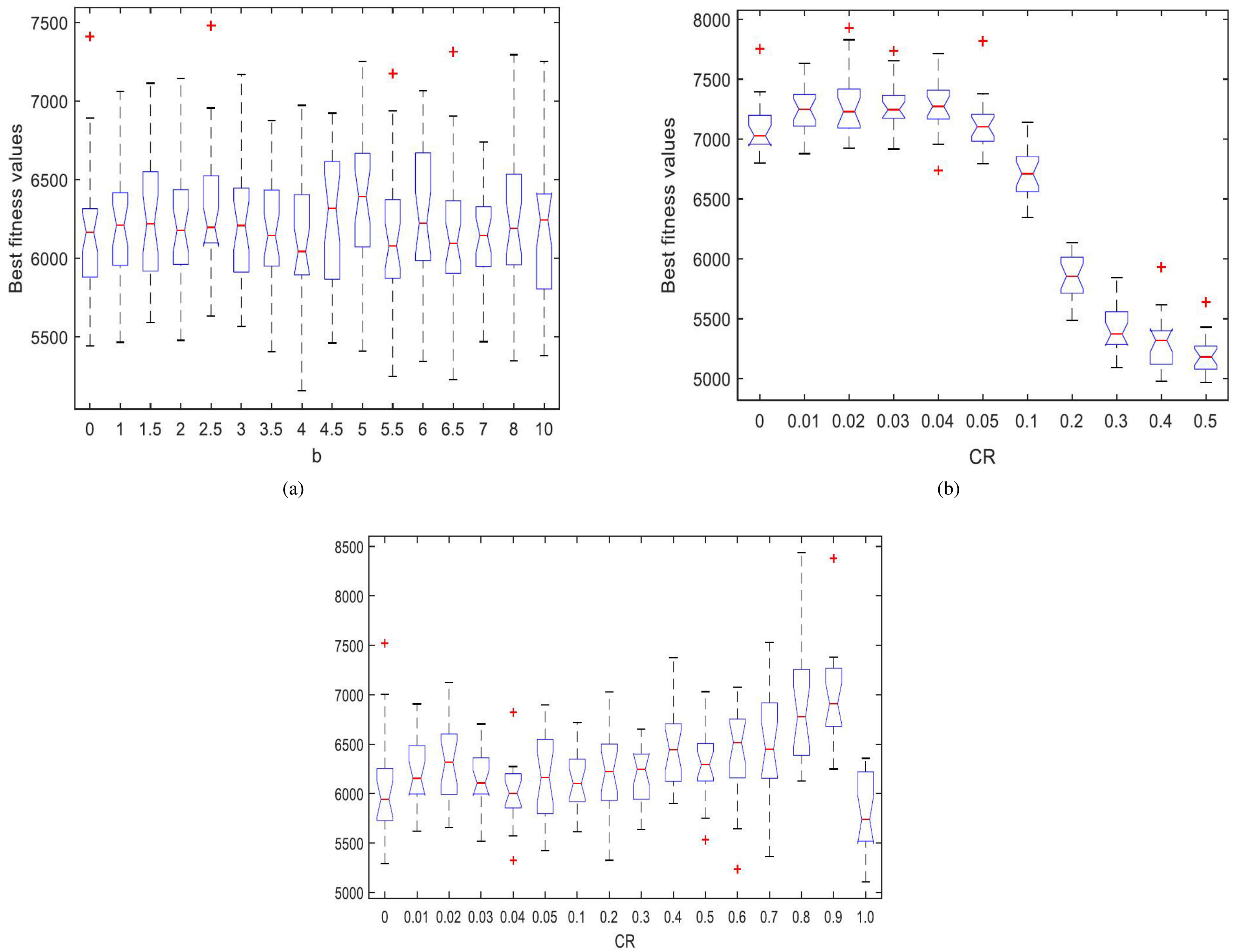
TABLE 5. Parameter setting for the proposed algorithm.

DWOA-LS		DEWOA		DES1/DES2	
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
Number of runs	30	Number of runs	30	Number of runs	30
Population size	30	Population size	30	Population size	30
The maximum number of iteration	5000	The maximum number of iteration	5000	The maximum number of iteration	5000
PALS2-many iterations ($subIter$)	100	CR	0.9	CR	0.02
Local Search Probability	0.7	B	5		

Lshade		WOA/LWOA/CWOA /SCA	
Parameter	**Value**	**Parameter**	**Value**
Number of runs	30	Number of runs	30
Population size	$18 \times N$	Population size	30
The maximum number of iteration	5000	The maximum number of iteration	5000
F	0.5	Chaotic map type for CWOA	Tent
CR	0.5	The Constant a of SCA	2.0
Min N	4	β in LWOA	1.5
Archive rate	1.4	$\vec{a}$ is linearly decreased by 2 to 0 for different variants of WOA	
p_best_rate	0.11		

it was clear that the change in the overlap score (fitness value) is quite significant when *subIter* is equal to 100 and become nearly constant when *subIter* is equal to 120 and 500. Consequently, the best value for *subIter* is 100. Note that, in the remaining experiments, the cutoff value is reset to 10. Table 5 presents the values of the DWOA-LS parameters and the other algorithms parameters used in the conducted experiments.

Regarding LSP, at the start, a value of 0.2 for LSP was selected randomly, and then the number of iterations used under this value was observed until reaching the optimal solution forthe six instances mentioned previously. As a result of observation, it is notified that the proposed algorithm needs a high number of iterations to reach the optimal value for those instances. Therefore, another value of 0.3 was used to determine the influence of this parameter on the performance of the algorithm; using this value the optimal solution was reached in fewer iterations in comparison to the previously observed value for those instances. For a value of 0.5, the algorithm reached the optimal values for the same instances in fewer iterations compared with the other two values. For a value of 0.7, the proposed algorithms could reach the optimal values for those instances in fewer iterations compared with the others. For a value of 1.0, the proposed approach reached the optimal values in a number of iterations similar to 0.7. Therefore, within our experiments, a value of 0.7 was used instead of 1.0 to avoid the time complexity problem.



(a) (b) (c)

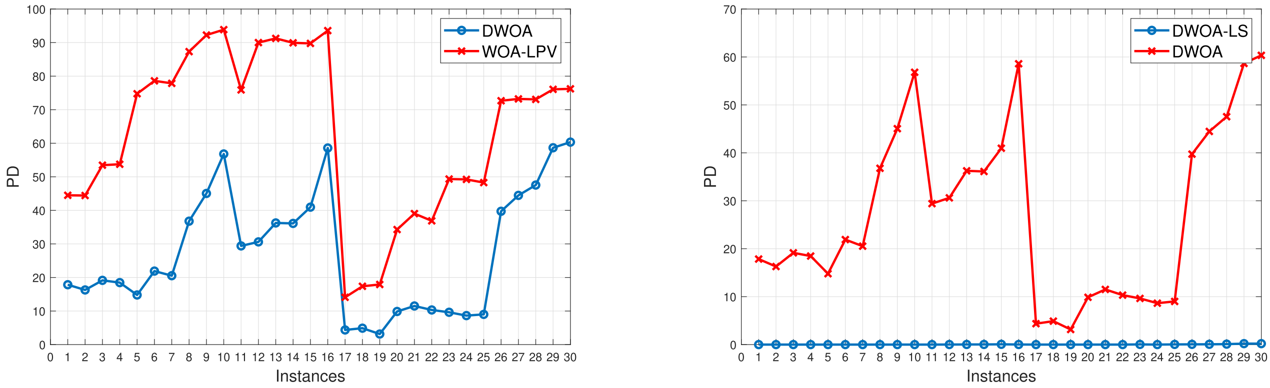
FIGURE 6. Depiction of tuning the parameters: (a) b; (b) CR of DE; and (c) CR of DEWOA.

Regarding the compared algorithms, five continuous WOA and DE variants and the sine-cosine algorithms mapped using LPV were compared with DWOA under the same number of iterations and population size assigned in Table 5:

- Improved Lévy flight whale optimization (LWOA) [68].
- Chaotic-based whale optimization algorithm (CWOA) [69].
- Differential evolution improved using differential evolution (DEWOA) [70].
- Differential evolution based on "DE/rand/1" scheme (DES1) [70].
- Differential evolution based on "DE/current to best/1" scheme (DES2) [70].
- Sine-cosine optimization algorithm (SCA) [71].

Because those algorithms were proposed for tackling continuous optimization problems rather than the discrete nature of DFAP, the parameters of those algorithms were tuned to determine the optimal relevant values for solving this problem. The standard WOA doesn't need any tuning for their parameters with the exception of parameter b that controls in the spiral shape. To adjust this parameter for the different variants of WOA, several values, involving 0, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 8, and 10, are randomly selected and experimented within 30 independent runs to determine the best value for this parameter. Based on our experimental results that is depicted in Fig.6(a), the best value for b is 5.

The performance of the differential evolution is based on two factors: scaling factor (F) that is here adapted as mentioned in [70] and the crossover rate (CR), so different experiments were separately performed to extract the best value of this parameter for three different variants of DE: DE based on the "DE/rand/1" scheme, DE based on the "DE/current to best/1" scheme, and the hybridized WOA and DE (DEWOA). For the DE based on the "DE/current to best/1" and "DE/rand/1", it is obvious for the outcomes depicted in Fig.6(b) that 0.02 is the best for CR. Similarly,



(a) (b)

FIGURE 7. Comparison of PD based on the overlap score of (a) DWOA and WOA-LPV; and (b) DWOA-LS and DWOA.

for DEWOA as depicted in Fig.6(c), the best value for CR for DEWOA is 0.9. Finally, SCA is a self-adaptive algorithm that has one parameter, control in the distance, which was set as declared in the cited paper.

C. PERFORMANCE OF DIFFERENT WOA AND DE VARIANTS

The purpose of this section is to assess the performance of the proposed algorithm: DWOA-LS through two main experiments:

- The first experiment compares the WOA integrated with LPV technique (WOA-LPV) and DWOA without using the local search method.
- The second observes the performance of different improved WOA and DE variants.
- The third experiment studies the effect of adding local search to the proposed algorithm by comparing DWOA and the proposed algorithm DWOA-LS.

1) THE COMPARISON BETWEEN DWOA AND WOA-LPV

This first experiment investigates the performance of two algorithms. The first algorithm is the standard WOA that uses the LPV technique (WOA-LPV). The second algorithm is DWOA without the local search method. This comparison is made to prove that the traditional mapping of continuous values to adapt WOA (WOA-LPV) for solving DFAP isn't effective. Table 6 presents the results obtained by the two algorithms based on the fitness function that uses the overlap score. The column opt shows the optimal overlap score for each benchmark instance obtained by the LinâĆŃ Kernighan heuristic (LKH) algorithm [72]. The best, average, and worst overlap score values are recorded in the table for running each algorithm 30 runs. By observing the results, DWOA attains much better results compared to WOA-LPV. For example, DWOA achieves higher overlap values for all the DFAP instances. The convergence of DWOA to the optimal solution is faster than WOA-LPV. This comparison demonstrates why WOA-LPV was not used and why the investigation used a discrete version of WOA (DWOA) that supports some operators from the evolutionary algorithms.

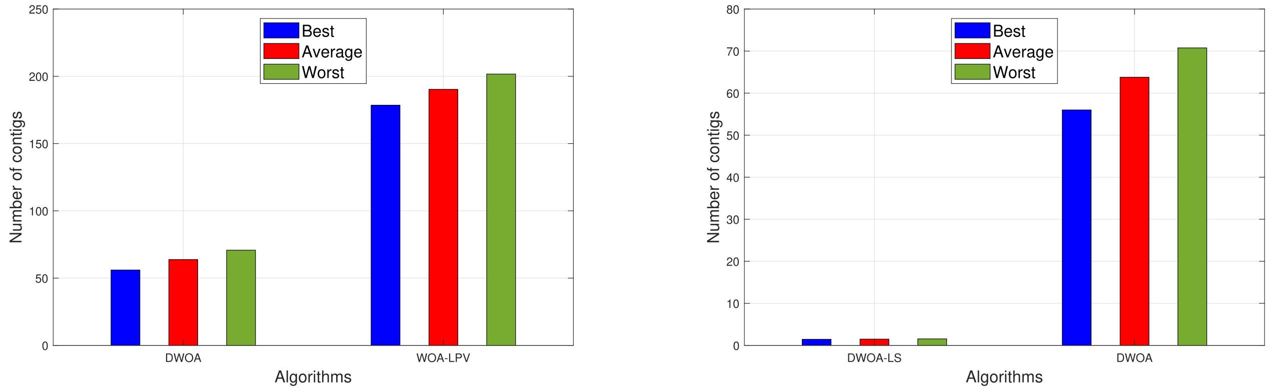
TABLE 6. A summary of the fitness values of DWOA and WOA-LPV.

2*Instances	2*Opt	DWOA			WOA-LPV		
		Best	Average	Worst	Best	Average	Worst
X4	11478	**10259**	**9430**	**8708**	6977	6371	5424
X5	14161	**12838**	**11855**	**10879**	8764	7870	6934
X6	18301	**15529**	**14798**	**13581**	9518	8516	7327
X7	21271	**17985**	**17342**	**16536**	10859	9839	8828
M5	38746	**34243**	**33017**	**31724**	11858	9781	7669
M6	48052	**39301**	**37526**	**36383**	11684	10282	8668
M7	55171	**45665**	**43843**	**42846**	13739	12217	10118
J7	116700	**75948**	**73776**	**69988**	16731	14811	13047
BX4	227920	**129983**	**125256**	**118771**	21719	17666	15089
BX7	445422	**205539**	**192391**	**184539**	30743	27369	22788
AC1	47618	**36159**	**33615**	**32259**	12849	11477	10726
AC2	151553	**108964**	**105164**	**100328**	18829	15155	12231
AC3	167877	**113856**	**107020**	**100069**	17747	14687	12516
AC5	163906	**111344**	**104715**	**100091**	18843	16501	14504
AC7	180966	**118834**	**106835**	**100272**	21710	18535	15632
AC9	344107	**152257**	**142521**	**133764**	26252	22130	19042
F305	596	**588**	**570**	**545**	536	512	479
F400	777	**764**	**739**	**706**	710	642	581
F500	921	**914**	**892**	**833**	804	756	708
F315	1581	**1470**	**1425**	**1362**	1121	1039	937
F412	1573	**1461**	**1392**	**1303**	1042	959	902
F498	1570	**1465**	**1408**	**1336**	1173	991	888
F307	2793	**2579**	**2524**	**2464**	1513	1415	1247
F415	2860	**2658**	**2613**	**2559**	1631	1452	1352
F512	2732	**2540**	**2486**	**2419**	1569	1413	1316
F508	18112	**11481**	**10916**	**10331**	5123	4950	4760
F635	22498	**13063**	**12494**	**11645**	6193	6026	5867
F737	25218	**13616**	**13229**	**12954**	7058	6791	6529
F1343	49042	**20751**	**20277**	**19713**	12058	11737	11563
F1577	57373	**23362**	**22754**	**22355**	13824	13643	13309

Also, Fig. 7 (a) depicts a comparison between DWOA and WOA-LPV in terms of the percentage deviation (PD). PD shows the percentage of the difference between the average fitness value found by an algorithm and the optimal fitness value divided by the optimal fitness value. We can calculate PD as:

$$PD = \frac{average - opt}{opt} \times 100 \tag{15}$$

Fig. 7 (a) shows that the DWOA is closer to the optimal solution than the standard version mapped using LPV. In LPV, there is no one-to-one mapping between the continuous solution and the permutation one because the permutation solution can be encoded by an infinite number of the continuous



(a) (b)

FIGURE 8. Comparison based on the mean of best, average and standard deviation of number of contigs between (a) DWOA and WOA-LPV; and (b) DWOA-LS and DWOA.

TABLE 7. A summary of the number of contigs obtained by the DWOA and WOA-LPV.

2*Instances	DWOA			WOA-LPV		
	Best	Average	Worst	Best	Average	Worst
X4	1.0	1.6	3.0	1.0	1.9	4.0
X5	1.0	2.33	5.0	1.0	3.3	6.0
X6	1.0	2.8	4.0	3.0	9.0	5.6
X7	1.0	2.4	4.0	2.0	4.13	8.0
M5	3.0	5.53	9.0	19.0	25.0	31.0
M6	7.0	10.4	14.0	30.0	40.53	57.0
M7	4.0	9.93	14.0	106	117	124
J7	18.0	27.0	23.0	82.0	93.3	107.0
BX4	17.0	22.0	29.0	83.0	98.6	114.0
BX7	40.0	47.2	55.0	187.0	202.4	227.0
AC1	47.0	51.33	56.0	126.0	136.233	145.0
AC2	6.0	11.8	16.0	63.0	79.9	95.0
AC3	16.0	21.46	28.0	107	123.3	137
AC5	16.0	23.4	33.0	140	158.6	179
AC7	23.0	30.13	38.0	185	198.0	229.0
AC9	37.0	49.86	60.0	214	242	264
F305	3.0	5.0	7.0	7.0	10.34	13.0
F400	3.0	5.2	7.0	9.0	10.43	13.0
F500	4.0	5.8	8.0	8.0	11.34	14.0
F315	6.0	7.93	10.0	15.0	21.86	26.0
F412	5.0	7.5	11.0	14.0	19.26	26.0
F498	3.0	5.0	9.0	13.0	19.76	25.0
F307	6.0	10.69	13.0	47.0	53.87	61.0
F415	8.0	10.53	13.0	46.0	51.76	61.0
F512	7.0	9.8	17.0	40.0	48.76	57.0
F508	110.0	126.46	138.0	361.0	374.96	388.0
F635	153.0	164.2	181.0	450.0	478.66	492.0
F737	169.0	193.03	206.0	539.0	561.2	584.0
F1343	394.0	438.26	473.0	1060	1076.4	1085.0
F1577	516.0	542.86	571.0	1221.0	1249.06	1275.0

numerical vectors. Because this disadvantage was solved in DWOA, it performs better than WOA-LPV.

However, the fitness values using the overlap score alone can't be used to assess the quality of the algorithm, because the algorithm may achieve high fitness values, but the obtained order of fragments contains a large number of contigs. Therefore, we compare the two algorithms based on the number of contigs attained from each algorithm, as recorded in Table 7. It can be observed that the results of DWOA outperform WOA-LPV based on the number of contigs for all instances. Based on these results, DWOA is better than WOA-LPV for solving the DFAP. DWOA outperform

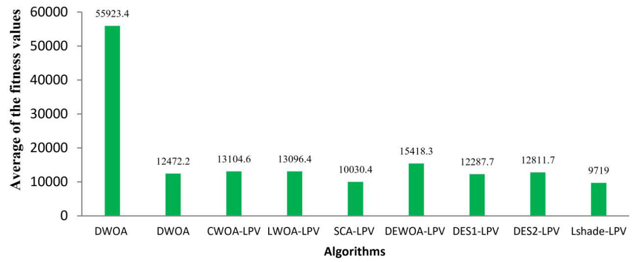
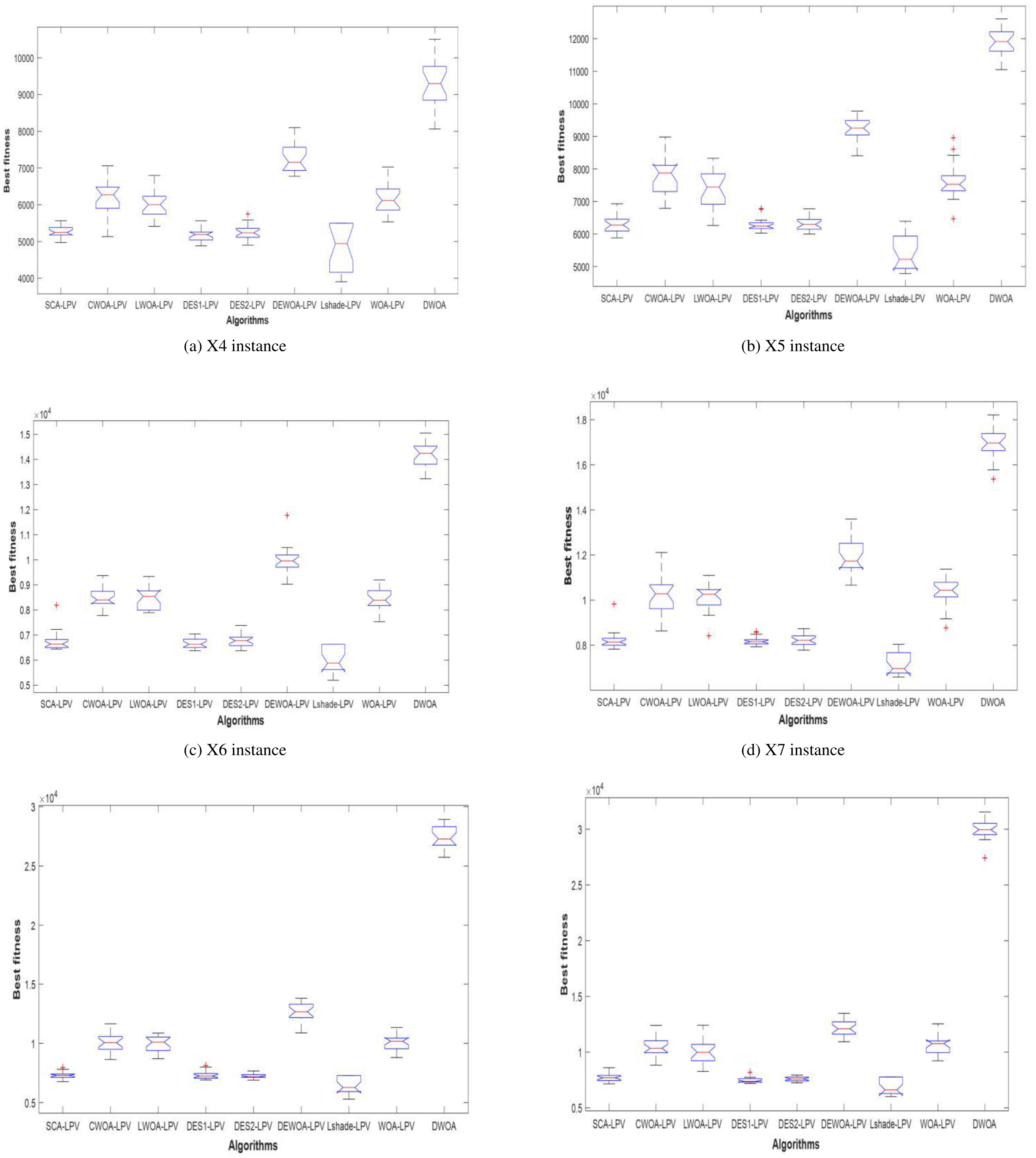


FIGURE 9. Average of the fitness values on X4-BX7 instances.

WOA-LPV in terms of the number of contigs and the overlap score. Fig. 8 (a) shows a comparison between DWOA and WOA-LPV based on the average number of contigs for all instances. For the best case of the algorithm, we can see that DWOA obtains in average 54.1contigs for all the instances, whereas WOA-LPV achieves 172.6 contigs. As can be seen from the Fig. 8 (a), the DWOA achieves the minimum number of contigs compared to WOA-LPV for the best, average, and worst cases.

2) COMPARISON OF DIFFERENT WOA AND DE VARIANTS

After completing the comparison between the DWOA and WOA-LPV, in this section, different robust WOA and DE variants, in addition to the SCA are compared with DWOA to prove its efficacy over the other recent improved variants on the instances from X4 to BX7. In the start, each variant is executed for 30 independent runs and the obtained outcomes in average are recorded in Table 8. Afterwards, those outcomes were observed to see the performance of DWOA; this observation shows that the DWOA could be superior to the other algorithms due to replacing the mapping phase by some genetic operators to get rid of the infinite number of the updated solutions that could represent the same permutation. In Fig. 9, the average of the outcomes recorded in Table 8 for each algorithm is graphically depicted to show more clear the superiority of DWOA. This figure shows



(a) X4 instance

(b) X5 instance

(c) X6 instance

(d) X7 instance

(e) M5 instance

(f) M6 instance

FIGURE 10. Boxplot obtained by different algorithms.

that DWOA outperforms the other algorithms with a value of 55923, while Lshade-LPV performs worst with an amount of 9719. Additionally, Fig. 10 measures the distribution of the outcomes based on five metrics: minimum, first quartile (Q2), median, third quartile (Q3), and maximum for the instances X4, X5, X6, X7, and M5. Again, this figure shows the superiority of the proposed DWOA for the five observed instances over the five metrics.

Then, the CPU time required by each algorithm is computed and recorded in Fig.11 which shows the increase in

TABLE 8. Comparison of different WOA and DE variants.

Instances	opt	Average Fitness Value								
		DWOA	WOA-LPV	CWOA-LPV [69]	LWOA-LPV [68]	SCA-LPV [71]	DEWOA-LPV [70]	DES1-LPV [70]	DES2-LPV [70]	LSHADE-LPV [73]
X4	11478	9430	6371	6260	6027	5262	7272	7161	7228	5234
X5	14161	11855	7870	7666	7617	6396	8968	8765	9008	6050
X6	18301	14798	8516	8584	8431	6711	9948	9471	9950	6599
X7	21271	17342	9839	10076	10173	8175	12036	11442	11923	7825
M5	38746	33017	9781	10142	10267	7273	12897	10546	11549	7275
M6	48052	37526	10282	10058	9953	7562	12281	9834	10721	7468
M7	55171	43843	12217	11680	12136	8876	14728	11459	12553	8703
J7	116700	73776	14811	15559	15839	11634	18081	13062	13627	10848
BX4	227920	125256	17666	20280	20364	15299	25011	17037	17511	14867
BX7	445422	192391	27369	30741	30157	23116	32961	24100	24047	22321

TABLE 9. Wilcoxon rank sum test.

Instances		Statistical rank sum test between the DWOA and those below algorithms							
		WOA-LPV	CWOA-LPV [69]	LWOA-LPV [68]	SCA-LPV [71]	DEWOA-LPV [70]	DES1-LPV [70]	DES2-LPV [70]	Lshade-LPV [73]
X4	p	6.78604E-08	6.79562E-08	6.78604E-08	6.79562E-08	7.89803E-08	6.79562E-08	6.78604E-08	6.78604E-08
	h	1	1	1	1	1	1	1	1
X5	p	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1
X6	p	6.78604E-08	6.79562E-08	6.79562E-08	6.78604E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1
X7	p	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1
M5	p	6.79562E-08	6.78604E-08	6.79562E-08	6.78604E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1
M6	p	6.78604E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1
M7	p	6.77647E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1
J7	p	6.78604E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.78604E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1
BX4	p	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.78604E-08	6.78604E-08
	h	1	1	1	1	1	1	1	1
BX7	p	6.79562E-08	6.78604E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08	6.79562E-08
	h	1	1	1	1	1	1	1	1

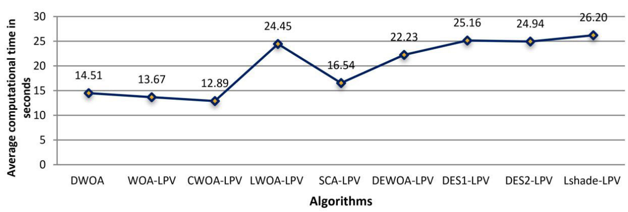


FIGURE 11. Average CPU time of each variant.

speed of the DWOA compare with others with the exception of WOA-LPV and CWOA-LPV that are very competitive with DWOA.

Finally, the Wilcoxon rank sum test [74] at a confidence level of 5% is used to show the significance of the DWOA over the other techniques. This test is based on two hypotheses: Null and alternative. This test assumes that there is no difference between the outcomes of a pair of the algorithms in the Null hypothesis case, in which the p-value is greater than the significant level (0.05) and h=0. Alternatively, it assumes that there is a difference between the two outcomes, in which case the p-value is less than the significant level (0.05) and h=1. Table 9 compares DWOA-LS with the other algorithms on the instances X4-BX7, showing that the alternative hypothesis is accepted with all the algorithms over all the instances and this confirms the significance of the DWOA over those algorithms.

3) COMPARISON OF DWOA AND DWOA-LS

The third experiment is conducted to study the effect of integrating the local search method with the proposed algorithm. We use the overlap score as a fitness function to evaluate the comparison between DWOA without Local Search (DWOA), and DWOA combined with the Local Search (DWOA-LS). Table 10 presented the results of the two algorithms for the best, average, and worst cases. Based on the results introduced, the DWOA-LS assembler finds the optimal solution for 20 out of 30 instances and can reach one contig for 22 out of 30 DFAP instances. From this analysis, the proposed algorithm DWOA-LS outperforms DWOA for all the DFAP instances. DWOA-LS obtained substantially better results for the medium and large DFAP cases, as opposed to the disappointing performance of DWOA. This superiority in the performance of DWOA-LS over DWOA is a result of the PMFC that enables DWOA to escape the local minima in which it may fall during the optimization process as a result of reducing the diversity among the members of the population. Subsequently, the possibility of reaching to other permutations that may improve the quality of the solutions is substantially reduced. In addition, this LS accelerates convergence toward the best-so-far solution because, applying it after the OC, generates the updated solutions based on the best-so-far position and the current one and this may increase convergence toward the optimal solution. Additionally, Fig. 7 (b) shows the percentage deviation between the two algorithms,

TABLE 10. A summary of the fitness results obtained by DWOA and DWOA-LS.

Instances	Opt	DWOA-LS			DWOA		
		Best	Average	Worst	Best	Average	Worst
X4	11478	**11478**	**11478**	**11478**	10259	9430	8708
X5	14161	**14161**	**14161**	**14161**	12838	11855	10879
X6	18301	**18301**	**18301**	**18301**	15529	14798	13581
X7	21271	**21271**	**21271**	**21271**	17985	17342	16536
M5	38746	**38746**	**38746**	**38746**	34243	33017	31724
M6	48052	**48052**	**48052**	**48052**	39301	37526	36383
M7	55171	**55171**	**55171**	**55171**	45665	43843	42846
J7	116700	**116700**	**116700**	**116700**	75948	73776	69988
BX4	227920	**227920**	**227920**	**227920**	129983	125256	118771
BX7	445422	**445422**	**445422**	**445422**	205539	192391	184539
AC1	47618	**47618**	**47618**	**47618**	36159	33615	32259
AC2	151553	**151546**	**151538**	**151528**	108964	105164	100328
AC3	167877	**167854**	**167838**	**167823**	113856	107020	100069
AC5	163906	**163869**	**163859**	**163853**	111344	104715	100091
AC7	180966	**180902**	**180865**	**180854**	118834	106835	100272
AC9	344107	**344076**	**344050**	**344035**	152257	142521	133764
F305	596	**596**	**596**	**596**	588	570	545
F400	777	**777**	**777**	**777**	764	739	706
F500	921	**921**	**921**	**921**	914	892	833
F315	1581	**1581**	**1581**	**1581**	1470	1425	1362
F412	1573	**1573**	**1573**	**1573**	1461	1392	1303
F498	1570	**1570**	**1570**	**1570**	1465	1408	1336
F307	2793	**2793**	**2792**	**2791**	2579	2524	2464
F415	2860	**2860**	**2860**	**2860**	2658	2613	2559
F512	2732	**2732**	**2731**	**2731**	2540	2486	2419
F508	18112	**18108**	**18103**	**18099**	11481	10916	10331
F635	22498	**22493**	**22484**	**22481**	13063	12494	11645
F737	25218	**25197**	**25194**	**25180**	13616	13229	12954
F1343	49042	**48951**	**48944**	**48936**	20751	20277	19713
F1577	57373	**57271**	**57260**	**57251**	23362	22754	22355

from which it can be observed that DWOA-LS has the lowest percentage deviation for all DFAP instances.

In addition to the comparison of the overlap between DWOA-LS and DWOA, Table 11 provides a comparison based on the number of contigs. The proposed DWOA-LS outperform DWOA in all instances. From Fig. 8 (b), we can see significant differences between the two algorithms in the average number of contigs for all DFAP instances. The average number of contigs in the best case for DWOA-LS is 1.433 contigs and for the DWOA it is 54.17 contigs. Furthermore, the average number of contigs in the worst case is 1.533 contigs and 68.5 for DWOA. It is obvious that using the local search affects significantly the obtained number of contigs and the overlap score. Broadly speaking, in the first subIter of iterations, the proposed approaches uses the LS, PALS2-many, to minimize the number of contigs even if the overlap score will be minimized. Then, after ending this number of iterations, another LS, abbreviated as PMFC, replaces PALS2-many with the objective of maximizing the fitness value (overlap score) while preserving or minimizing the current number of contigs. The LS within the optimization process therefore plays a double role: the first is to arbitrarily minimize the number of contigs within the first *subIter* of the optimization process, while the second plays a significant role in improving the overlap score while preserving or minimizing the number of contigs.

From the experiments conducted in this section, the following conclusions can be drawn:

- Simulating the behaviors of the WOA by borrowing some genetic operators can significantly improve its performance as a result of utilizing effectively the whole optimization process and the individuals within the population by erasing the problems of the traditional mapping methods that may generate the same permutation by different real-value positions.
- The experiments shows the superiority of DWOA over the recent robust algorithms mapped using the LPV.
- Then, to accelerate convergence, two-phase-based LS is integrated with the DWOA: in the first phase, an LS known as PALS2-many is applied within the first subIter of iterations focused only on minimizing the number of contigs, while the second phase is applied after subIter iterations to preserve or minimize the current number of contigs while maximizing the overlap score.
- The experiments show that PALS2-many and PMFC enhance the performance of DWOA.
- Applying PALS2-many in the first specified number of iterations concentrating on achieving a one-contig solution (in addition to applying PMFC within the remaining iterations focusing on the maximum overlap while preserving or decreasing the number of contig) assists in the production of more promising solutions that can attain the optimal overlap with a one-contig solution in most instances.

TABLE 11. A summary of the number of contigs between DWOA-LS and DWOA.

Instances	DWOA-LS			DWOA		
	Best	Average	Worst	Best	Average	Worst
X4	1.0	1.0	1.0	1.0	1.6	3.0
X5	1.0	1.0	1.0	1.0	2.33	5.0
X6	1.0	1.0	1.0	1.0	2.8	4.0
X7	1.0	1.0	1.0	1.0	2.4	4.0
M5	1.0	1.0	1.0	3.0	5.53	9.0
M6	1.0	1.0	1.0	7.0	10.4	14.0
M7	1.0	1.0	1.0	4.0	9.93	14.0
J7	1.0	1.0	1.0	18.0	27.0	23.0
BX4	1.0	1.0	1.0	17.0	22.0	29.0
BX7	1.0	1.0	1.0	40.0	47.2	55.0
AC1	2.0	2.0	2.0	47.0	51.33	56.0
AC2	2.0	2.0	2.0	6.0	11.8	16.0
AC3	2.0	2.0	2.0	16.0	21.46	28.0
AC5	2.0	2.0	2.0	16.0	23.4	33.0
AC7	2.0	2.0	2.0	23.0	30.13	38.0
AC9	7.0	7.0	7.0	37.0	49.86	60.0
F305	1.0	1.2	2.0	3.0	5.0	7.0
F400	2.0	2.0	2.0	3.0	5.2	7.0
F500	2.0	2.8	3.0	4.0	5.8	8.0
F315	1.0	1.0	1.0	6.0	7.93	10.0
F412	1.0	1.2	2.0	5.0	7.5	11.0
F498	1.0	1.0	1.0	3.0	5.0	9.0
F307	1.0	1.0	1.0	6.0	10.69	13.0
F415	1.0	1.0	1.0	8.0	10.53	13.0
F512	1.0	1.0	1.0	7.0	9.8	17.0
F508	1.0	1.0	1.0	110.0	126.46	138.0
F635	1.0	1.0	1.0	153.0	164.2	181.0
F737	1.0	1.0	1.0	169.0	193.03	206.0
F1343	1.0	1.0	1.0	394.0	438.26	473.0
F1577	1.0	1.0	1.0	516.0	542.86	571.0

D. COMPARISON BETWEEN THE PROPOSED ASSEMBLER AND THREE RECENT ASSEMBLERS

This section is concerned with investigating the superiority of DWOA-LS over other existing assemblers. The experiments

TABLE 12. A comparison of fitness and number of contigs between DWOA-LS and other assemblers.

2*Instances	DWOA-LS		CSA-P2M*Fit		P2M*Fit		GA-P2M*Fit	
	Fitness	NC	Fitness	NC	Fitness	NC	Fitness	NC
X4	**11478**	**1.0**	**11478**	**1.0**	11478	**1.0**	**11478**	**1.0**
X5	**14161**	**1.0**	**14161**	**1.0**	14157	2.0	**14161**	**1.0**
X6	**18301**	**1.0**	**18301**	**1.0**	18301	**1.0**	**18301**	**1.0**
X7	**21271**	**1.0**	**21271**	**1.0**	21271	**1.0**	**21271**	**1.0**
M5	**38746**	**1.0**	**38746**	3.0	38661	5.0	**38746**	3.0
M6	**48052**	**1.0**	**48052**	2.0	48052	2.0	**48052**	2.0
M7	**55171**	**1.0**	**55171**	2.0	55169	3.0	**55171**	2.0
J7	**116700**	**1.0**	**116700**	2.0	116352	4.0	**116700**	2.0
BX4	**227920**	**1.0**	**227920**	9.0	226858	15.0	227920	8.0
BX7	**445422**	**1.0**	**445422**	2.0	442708	10.0	445422	2.0
AC1	**47618**	**2.0**	**47618**	6.0	47140	10.0	47609	5.0
AC2	**151546**	**2.0**	151545	61.0	151256	90.0	151520	49.0
AC3	167854	**2.0**	**167861**	74.0	167025	113.0	167781	59.0
AC5	163869	**2.0**	**163891**	82.0	163107	144.0	163758	80.0
AC7	180902	**2.0**	**180924**	86.0	179822	159.0	180748	81.0
AC9	344076	**7.0**	**344078**	64.0	343059	124.0	343968	54.0
F305	**596**	**1.0**	**596**	19.0	596	19.0	596	19.0
F400	**777**	**2.0**	**777**	14.0	777	14.0	777	14.0
F500	**921**	**2.0**	**921**	14.0	921	14.0	921	14.0
F315	**1581**	**1.0**	**1581**	26.0	1575	26.0	1581	26.0
F412	**1573**	**1.0**	**1573**	26.0	1568	26.0	1573	26.0
F498	**1570**	**1.0**	**1570**	28.0	1568	28.0	1570	28.0
F307	**2793**	**1.0**	**2793**	69.0	2772	69.0	2793	69.0
F415	**2860**	**1.0**	**2860**	65.0	2843	65.0	2860	65.0
F512	**2732**	**1.0**	2732	69.0	2713	69.0	2732	69.0
F508	18108	**1.0**	**18110**	261.0	17885	260.0	18082	259.0
F635	**22493**	**1.0**	22492	333.0	22119	335.0	22454	332.0
F737	25197	**1.0**	**25206**	403.0	24793	406.0	25163	403.0
F1343	48951	**1.0**	**49012**	644.0	47956	663.0	48834	641.0

evaluate the performance of our proposed assemblers with the other assemblers based on three performance measures:

1) The fitness function using the overlap score.
2) The number of contigs.
3) An evaluation function called $F + C$.

The third experiment compares DWOA-LS and other three assemblers based on their outcomes in the published papers, including CSA-P2M*Fit [35], P2M*Fit [35], and GA-P2M*Fit [35]. The fitness function depending on the best overlap score and the minimum number of contigs (NC) are used as two main performance measures in this experiment. Table 12 introduces the results of the four algorithms. Although the two assemblers DWOA-LS and CSA-P2M*Fit find the optimal fitness values for 20 out of 30 instances, DWOA-LS outperforms CSA-P2M*Fit. DWOA-LS obtains a one-contig solution that has the optimal overlap score in 17 DFAP datasets, but CSA-P2M*Fit and GA-P2M*Fit achieve this solution in only four DFAP datasets. In contrast to DWOA-LS in the other assemblers, when the DFAP becomes larger, the number of contigs becomes disastrous. The maximum number of contigs is seven contigs for DWOA-LS, while the maximum number of contigs for the other assemblers is 788 for CSA-P2M*Fit and GA-P2M*Fit and 810 for P2M*Fit.

Graphically, Fig. 12 compares the different assemblers based on the PD values to determine how close each algorithm is to the optimal overlap score. As can be seen from the Fig. 12, the overlap score of the CSA-P2M*Fit is slightly higher in some instances. However, the superiority of the CSA-P2M*Fit in the overlap score does not mean that it is better because in some cases, a solution that has a better fitness value may generate a larger number of contigs. Therefore, the difference between our algorithm and the CSA-P2M*Fit in the overlap score doesn't provide a useful evaluation of performance for solving DFAP. In this paper, another evaluation function is proposed to evaluate the performance of the assemblers illustrated below.

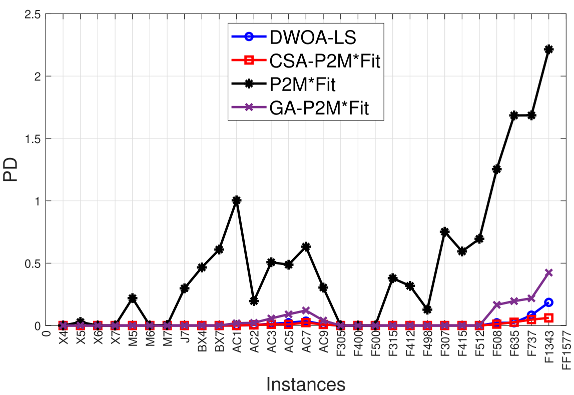


FIGURE 12. A comparison of PD for DWOA-LS with other three assemblers based on the best overlap score.

As mentioned earlier, the fitness using the overlap score alone cannot be used to judge the quality of the assemblers because, in some situations, a solution that has a better fitness

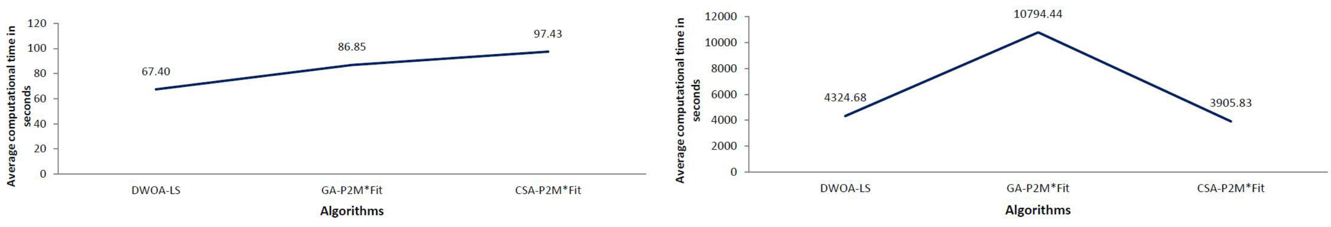


FIGURE 13. Comparison of the CPU values for DWOA-LS, CSA-P2M*Fit, and GA-P2M*Fit 921 on (a) instance X4-BX7; and (b) the reminder of the instances.

TABLE 13. A comparison among algorithms based on F+C function.

Instances	Optimal	P2M*Fit	GA-P2M*Fit	CSA-P2M*Fit	DWOA-LS
X4	2	1.987	**2**	**2**	**2**
X5	2	1.947	1.995	1.981	**2**
X6	2	1.972	**2**	**2**	**2**
X7	2	1.974	**2**	**2**	**2**
M5	2	1.94	1.984	1.984	**2**
M6	2	1.986	1.994	1.989	**2**
M7	2	1.976	1.994	1.989	**2**
J7	2	1.973	1.996	1.997	**2**
BX4	2	1.954	1.983	1.981	**2**
BX7	2	1.971	1.994	1.998	**2**
AC1	2	1.942	1.984	1.980	**1.996**
AC2	2	1.781	1.888	1.859	**1.997**
AC3	2	1.785	1.896	1.871	**1.998**
AC5	2	1.785	1.888	1.878	**1.999**
AC7	2	1.798	1.901	1.898	**1.999**
AC9	2	1.866	1.945	1.935	**1.994**
F305	2	1.273	1.28	1.28	**1.992**
F400	2	1.474	1.48	1.48	**1.96**
F500	2	1.513	1.518	1.518	**1.933**
F315	2	1.49	1.5	1.5	**2**
F412	2	1.485	1.5	1.5	**1.996**
F498	2	1.45	1.46	1.46	**2**
F307	2	1.302	1.32	1.319	**1.999**
F415	2	1.341	1.36	1.359	**2**
F512	2	1.305	1.32	1.319	**1.999**
F508	2	1.465	1.489	1.487	**1.999**
F635	2	1.447	1.476	1.476	**1.999**
F737	2	1.421	1.451	1.453	**1.999**
F1343	2	1.476	1.517	1.518	**1.998**
F1577	2	1.453	1.495	1.498	**1.998**

using overlap score can have a large number of contigs. Therefore, the judgment has to be based on two factors:

1) The primary factor is minimizing the number of contigs with the target of reaching one contig.
2) The second factor is maximizing the overlap score.

As a result, a new fitness function is proposed to evaluate the performance of the assemblers based on the previous two factors. This fitness function is called (F+C), which is calculated according to the following formula:

$$F + C = \frac{avg\_fitness}{opt} + \frac{NF - NC + 1}{NF} \quad (16)$$

where *avg_fitness* is the average obtained overlap score for a given algorithm, and opt is the best known overlap score. NF and NC represent the number of fragments and contigs, respectively.

Based on the results introduced in Table 13, we can see the superiority of the DWOA-LS for all instances for F+C values. Our algorithm contributes significantly to reduce the number of contigs and to increase the overlap score among the fragments compared with the other assemblers. DWOA-LS can reach the ultimate objective that contains a one-contig solution with the optimal overlap score in 13 datasets. For the remaining datasets, DWOA-LS is too close to obtain 2. CSA-P2M*Fit and GA-P2M*Fit get the optimal solution in only three datasets, as shown in Table 13.

To measure the CPU time for DWOA-LS, CSA-P2M*Fit, and GA-P2M*Fit, the latter two algorithms were implemented to make a fair comparison between the CPU time consumed by each. The investigation of CPU time was divided into two experiments. The first computed the CPU time for the first benchmark until reaching the optimal value for each instance. After running each assembler (DWOA, GA-P2M*Fit, and CSA-P2M*Fit) for 30 independent runs, the average was calculated of the computational time needed for each of the datasets from X4 to BX7. As can be seen in Fig. 13a, the proposed assembler is faster than CSA-P2M*Fit and GA-P2M*Fit in reaching the optimal solution for those datasets.

The second experiment computed CPU time for the other datasets, which includes some instances with an optimal solutions that the algorithms couldn't reach, which investigates the CPU time consumed by each one until the end of the optimization process. After running each algorithm 30 independent on those datasets, the average CPU time is introduced in Fig. 13b, which shows that CSA-P2M*Fit is faster than the proposed algorithm and GA-P2M*Fit. However, the proposed algorithm is very close to the CPU time consumed by CSA-P2M*Fit, so they are competitive in terms of the CPU time.

To complete this experiment a statistical test known as the statistical ranking color scheme (SRCS) [75] is used to compare the algorithms: DWOA-LS, CSA-P2M*Fit, GA-P2M*Fit and P2M*Fit. SRCS sets all the algorithms to an initial value of 0. Then, the Krusskal–Wallis test is employed for detecting whether there are any differences between the algorithms. If there is no difference, the algorithms terminate in their initial value 0. If there is a significant difference among the algorithms, the Mann–Whitney–Wilcoxon + Holm test is applied for each possible pair of the algorithms, and the ranking value of the algorithm with the highest performance is incremented by 1, and the other is decreased by 1. If there are no differences, then the ranking value is preserved. Hence, the top-ranked algorithm has the highest performance. Here, this test is applied on the proposed algorithm and another three assemblers to illustrate the superiority of our proposed algorithm.

TABLE 14. The SRCS ranking results of different algorithms based on the average fitness value/contigs.

Instance	DWOA-LS	CSA-P2M*Fit	P2M*Fit	GA-P2M*Fit
X4	1/0	1/0	-3/0	1/0
X5	1/3	1/-1	-3/-3	1/1
X6	1/1	1/1	-3/3	1/1
X7	1/1	1/1	-3/-3	1/1
M5	1/3	1/0	-3/-3	1/0
M6	1/3	1/-1	-3/-3	1/1
M7	2/3	2/-1	-3/-3	1/1
J7	2/3	2/0	-3/-3	-1/0
BX4	2/3	2/-1	-3/-3	-1/1
BX7	2/3	2/1	-3/-3	-1/-1
AC1	2/3	2/3	-3/-3	-1/1
AC2	3/3	1/-1	-3/-3	-1/1
AC3	1/3	3/-1	-3/-3	-1/1
AC5	1/3	3/-1	-3/-3	-1/1
AC7	1/3	3/-1	-3/-3	-1/1
AC9	1/3	3/-1	-3/-3	-1/1
F305	1/3	1/-1	-3/-1	1/-1
F400	1/3	1/-1	-3/-1	1/-1
F500	1/3	1/-1	-3/-1	1/-1
F315	1/3	1/0	-3/-3	1/0
F412	1/3	1/0	-3/-3	1/0
F498	1/3	1/0	-3/-3	1/0
F307	0/3	0/0	-3/-3	3/0
F415	0/3	0/0	-3/-3	3/0
F512	0/3	0/0	-3/-3	3/0
F508	1/3	3/-1	-3/-3	-1/1
F635	3/3	1/-1	-3/-3	-1/1
F737	1/3	3/0	-3/-3	-1/0
F1343	1/3	3/-1	-3/-3	-1/1
F1577	1/3	3/-1	-3/-3	-1/1

Table 14 shows the ranking value for four different assemblers according to the average fitness value-based the average overlap and the average number of contigs obtained by each algorithm for each instance. Based on the ranking values introduced, our proposed algorithm outperforms all other algorithms based on the number of contigs in all instances. If the algorithm attains a value 0, it means that all the algorithms obtain the same results. If the algorithm achieves any value of (1, 2, 3), it means that the algorithm outperforms one, or two, or three algorithms, respectively. If the algorithm obtains any value of (−1, −2, −3), it means that there are one, or two, or three algorithms that precede this algorithm, respectively. DWOA-LS achieves a value of 3 in most of the instances for the number of contigs. Also, for the fitness value, DWOA-LS is equivalent to the other algorithms or outperforms one or more of three algorithms. CSA-P2M*fit outperforms our proposed algorithm in 8 instances based on the average fitness value and so is a little higher in fitness value. There is, however, a clear difference in the number of contigs between the two algorithms as the proposed algorithm produces an output that is closer to the optimal order of the fragments.

E. COMPARISON OF THE PROPOSED ASSEMBLER AND OTHER ASSEMBLERS

The fourth experiment compared the proposed assembler with other selected state-of-the-art assemblers:

1) Transposition restarting and recentering genetic algorithm with island model (Trans. RRGA+IM) [50];
2) Problem aware local search (PALS) [53];

TABLE 15. The SRCS ranking results of the algorithms in terms of F+C values.

Instance	DWOA-LS	CSA-P2M*Fit	GA-P2M*Fit	P2M*Fit
X4	1	1	1	-3
X5	3	-1	1	-3
X6	1	1	1	-3
X7	1	1	1	-3
M5	3	0	0	-3
M6	3	-1	1	-3
M7	3	-1	1	-3
J7	3	1	-1	-3
BX4	3	-1	1	-3
BX7	3	1	-1	-3
AC1	3	-1	1	-3
AC2	3	-1	1	-3
AC3	3	-1	1	-3
AC5	3	-1	1	-3
AC7	3	-1	1	-3
AC9	3	-1	1	-3
F305	3	0	0	-3
F400	3	0	0	-3
F500	3	0	0	-3
F315	3	0	0	-3
F412	3	0	0	-3
F498	3	0	0	-3
F307	3	-1	1	-3
F415	3	-1	1	-3
F512	3	-1	1	-3
F508	3	-1	1	-3
F635	3	0	0	-3
F737	3	1	-1	-3
F1343	3	1	-1	-3
F1577	3	1	-1	-3

3) Parallel hybrid particle swarm optimization and deferential evolution (PPSO+DE) [56];
4) Firefly algorithm (FF) [59];
5) Genetic algorithm (GA) [64]
6) Queen-bee evaluation based on genetic algorithm (QEGA) [76]; and
7) Simulated annealing (SA) [76].

The comparison is based on the best of the overlap scores obtained on the first two DFAP collections (GenFrag and DNAgen). Table 16 records the results of the nine algorithms, from which it can be seen that DWOA-LS outperforms all other algorithms in all instances by obtaining the optimal overlap score values in eleven DFAP instances. SA is second bestwith three datasets. From the last columns at Table 16, which presents a comparison of the total average overlap among the algorithms over the first DFAP collections (GenFrag and DNAgen). The Average of the second column represents the average of all the optimal values recorded for these two collections, which is 128328. DWOA-LS performs best with a value of 128318, which is very close to 128328. This experiment shows that our proposed algorithm is robust and successful in tackling DFAP. Trans. RRGA+IM performs second best with a value of 127875. GA performs worst with a value of 119176.

F. SUMMARY OF OUR EXPERIMENTS

From the previous experiments, the proposed algorithm DWOA-LS has been shown to be an effective assembler for tackling DFAP compared to other existing assemblers. The proposed DWOA-LS is capable of obtaining the minimum

TABLE 16. A summary of the overlap score for DWOA-LS and other selected algorithms.

Instances	opt	Trans.RRGA+IM	PALS	PPSO+DE	FF	GA	QEGA	SA	DWOA-LS
X4	11478	**11478**	**11478**	**11478**	**11478**	**11478**	11476	**11478**	**11478**
X5	14161	14161	14021	13642	14075	13502	14027	14027	**14161**
X6	18301	**18301**	**18301**	**18301**	18097	17688	18266	**18301**	**18301**
X7	21271	21245	21210	20921	20898	20884	21208	**21271**	**21271**
M5	38746	38690	38528	38686	37743	37714	38578	38583	**38746**
M6	48052	48048	48048	47669	47033	46949	47882	48048	**48052**
M7	55171	55168	55067	54891	51509	52695	55020	55048	**55171**
J7	116700	116502	115320	114381	108701	111103	116222	116257	**116700**
BX4	227920	227297	225783	224797	211654	220029	227252	226538	**227920**
BX7	445422	442452	438215	429338	413630	416414	443600	436739	**445422**
AC1	47618	47477	46876	47264	45160	45565	47115	46955	**47618**
AC2	151553	151335	144634	147429	147460	143444	144133	144705	**151546**
AC3	167877	167268	156776	163965	164652	154947	156138	156630	**167854**
AC5	163906	163246	146591	161511	162915	145332	144541	146607	**163869**
AC7	180966	180033	158004	180052	179913	155873	155322	157984	**180902**
AC9	344107	343314	325930	335522	333815	313203	322768	324559	**344072**
Average	128328	127876	122799	125615	123046	119176	122722	122733	**128318**

number of contigs while increasing the overlap score. Also, DWOA-LS is proved to be more robust for solving DFAP as it performs well for medium- and large-scale instances. A new evaluation function has been proposed to measure the performance of the different assemblers based on achieving a one-contig solution and attaining a high overlap score. This function can be useful in situations when an algorithm gets a higher overlap, but the number of contigs is large. So, the best algorithm balances the two objectives.

VII. CONCLUSION AND FUTURE WORK

In this paper, a WOA was adapted to solve a discrete fragment assembly problem (DFAP). To fit this WOA for discrete problems (DWOA), the swap-based best-position mutation operator was used to simulate the action of encircling the prey to move the whale around prey within a shrinking circle. The ordered crossover operators were employed to simulate the spiral shape, where DWOA selects a random block of positions from the prey and his block is copied to the same locations in the current whale. Finally, to search for the prey, the whale positions were generated randomly from the fragment numbers instead of using a random whale to prevent the reduction of the variation in the population. A local search approach called PALS2-many was also employed with the proposed DWOA in a version abbreviated as DWOA-LS for a better order of fragments. The local search helps DWOA to minimize the number of contigs, in addition to maximizing the overlap score among the fragments. We propose a new evaluation function F+C to assess the quality of different assemblers. DWOA-LS was validated on 30 benchmark instances and compared with a number of the robust recent state-of-the-arts algorithm for the DFAP under two experiments. In the first experiment, DWOA was compared with five WOA and DE variants, in addition to SCA to demonstrate the superiority of DWOA to convert the continuous behaviors of the whale to discrete. Additionally, to show the significance of the DWOA, the Wilcoxon rank sum test was used to show the significance of DWOA over those algorithms. The second experiment was performed to show the superior performance of DWOA-LS over a number of recent robust state-of the arts assemblers suggested for the DFAP. The experimental results and statistical analyses of this experiment show that the DWOA-LS outperforms significantly the different assemblers in terms of the number of contigs, whilst being competitive for the overlap score with CSA-P2M*Fit, and superior to P2M*Fit and GA-P2M*Fit. Finally, DWOA-LS is shown to be the best approach. Despite its superiority, the proposed algorithm did not achieve better overlap scores than CSA-P2M*Fit on some instances, which is a limitation of the proposed approach, in addition to its time complexity.

Future work aims to apply DWOA to other existing problems such as travelling salesman problem, task scheduling, and the knapsack problem. Additionally, a new evaluation function can be considered for tackling DFAP to judge and guide the solutions inside the search space. Parallelization of the proposed algorithm can also achieve better results and to exploit the processing power of new computers.

CONFLICT OF INTEREST

Authors declare that there is no conflict of interest about the research.

FUNDING

This research has no funding source.

ETHICAL APPROVAL

This article does not contain any studies with human participants or animals performed by any of the authors.

REFERENCES

[1] G. Phillips-Wren, P. Sharkey, and S. M. Dy, "Mining lung cancer patient data to assess healthcare resource utilization," *Expert Syst. Appl.*, vol. 35, no. 4, pp. 1611–1619, Nov. 2008.

[2] N. Maleki, Y. Zeinali, and S. T. A. Niaki, "A k-NN method for lung cancer prognosis with the use of a genetic algorithm for feature selection," *Expert Syst. Appl.*, vol. 164, Feb. 2021, Art. no. 113981.

[3] Q. Wang, Y. Zhou, W. Zhang, Z. Tang, and X. Chen, "Adaptive sampling using self-paced learning for imbalanced cancer data pre-diagnosis," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113334.

[4] Y.-M. D. Lo, R. W. K. Chiu, R. W. Y. Chan, and L. S. Tam, "Sequencing analysis of circulating DNA to detect and monitor autoimmune diseases," U.S. Patent 10 174 375, Jan. 8, 2019.

[5] M. A. Farley, *Forensic DNA Technology*. Boca Raton, FL, USA: CRC Press, 2019.

[6] A. O. Amankwaa and C. McCartney, "The effectiveness of the UK national DNA database," *Forensic Sci. Int., Synergy*, vol. 1, pp. 45–55, Jan. 2019.

[7] A. Ghosh, S. Basu, H. Khatri, K. Chandra, and M. Thakur, "Ascertaining species of origin from confiscated meat using DNA forensics," *Mitochondrial DNA B*, vol. 4, no. 1, pp. 329–331, Jan. 2019.

[8] G. C. Oatley and B. W. Ewart, "Crimes analysis software: 'Pins in maps', clustering and Bayes net prediction," *Expert Syst. Appl.*, vol. 25, no. 4, pp. 569–588, Nov. 2003.

[9] Z. J. Li, Y. H. Chen, D. J. Zhang, G. F. Zhang, and B. X. Lu, "Genetic diversity analysis and DNA fingerprinting of the main japonica rice varieties in heilongjiang province," *Qual. Assurance Saf. Crops Foods*, vol. 11, no. 1, pp. 23–29, Feb. 2019.

[10] R. Lal, *Genetic Engineering of Plants for Crop Improvement*. Boca Raton, FL, USA: CRC Press, 2020.

[11] A. M. Anter, A. E. Hassenian, and D. Oliva, "An improved fast fuzzy c-means using crow search optimization algorithm for crop identification in agricultural," *Expert Syst. Appl.*, vol. 118, pp. 340–354, Mar. 2019.

[12] J. Kubalik, P. Buryan, and L. Wagner, "Solving the DNA fragment assembly problem efficiently using iterative optimization with evolved hypermutations," in *Proc. 12th Annu. Conf. Genet. Evol. Comput. (GECCO)*, 2010, pp. 213–214.

[13] Z. Ezziane, "Applications of artificial intelligence in bioinformatics: A review," *Expert Syst. Appl.*, vol. 30, no. 1, pp. 2–10, Jan. 2006.

[14] M. Abdel-Basset, V. Chang, and R. Mohamed, "HSMA_WOA: A hybrid novel slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest X-ray images," *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106642.

[15] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: A new method for stochastic optimization," *Future Gener. Comput. Syst.*, vol. 111, pp. 300–323, Oct. 2020.

[16] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications," *IEEE Trans. Ind. Informat.*, early access, Jun. 11, 2020, doi: 10.1109/TII.2020.3001067.

[17] M. Abdel-Basset, R. Mohamed, M. Elhoseny, R. K. Chakrabortty, and M. Ryan, "A hybrid COVID-19 detection model using an improved marine predators algorithm and a ranking-based diversity reduction strategy," *IEEE Access*, vol. 8, pp. 79521–79540, 2020.

[18] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine predators algorithm: A nature-inspired metaheuristic," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113377.

[19] E. H. Houssein, M. E. Hosney, D. Oliva, W. M. Mohamed, and M. Hassaballah, "A novel hybrid harris hawks optimization and support vector machines for drug design and discovery," *Comput. Chem. Eng.*, vol. 133, Feb. 2020, Art. no. 106656.

[20] Q. Askari, M. Saeed, and I. Younas, "Heap-based optimizer inspired by corporate rank hierarchy for global optimization," *Expert Syst. Appl.*, vol. 161, Dec. 2020, Art. no. 113702.

[21] I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, "Gradient-based optimizer: A new metaheuristic optimization algorithm," *Inf. Sci.*, vol. 540, pp. 131–159, Nov. 2020.

[22] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[23] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark generator for CEC 2009 competition on dynamic optimization," Univ. Leicester, Univ. Birmingham, Birmingham, U.K., Tech. Rep., 2008.

[24] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technol. Univ., Singapore, Tech. Rep., 2005.

[25] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.

[26] M. Abdel-Basset, R. Mohamed, S. Mirjalili, R. K. Chakrabortty, and M. J. Ryan, "Solar photovoltaic parameter estimation using an improved equilibrium optimizer," *Sol. Energy*, vol. 209, pp. 694–708, Oct. 2020.

[27] A. G. Hussien, A. E. Hassanien, E. H. Houssein, M. Amin, and A. T. Azar, "New binary whale optimization algorithm for discrete optimization problems," *Eng. Optim.*, vol. 52, no. 6, pp. 945–959, Jun. 2020.

[28] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, Oct. 2017.

[29] R. K. Agrawal, B. Kaur, and S. Sharma, "Quantum based whale optimization algorithm for wrapper feature selection," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106092.

[30] M. Abdel-Basset, D. El-Shahat, and A. K. Sangaiah, "A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 3, pp. 495–514, Mar. 2019.

[31] J. Nasiri and F. M. Khiyabani, "A whale optimization algorithm (WOA) approach for clustering," *Cogent Math. Statist.*, vol. 5, no. 1, Jun. 2018, Art. no. 1483565.

[32] M. Abdel-Basset, D. El-Shahat, and I. El-Henawy, "A modified hybrid whale optimization algorithm for the scheduling problem in multimedia data objects," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 4, p. e5137, Feb. 2020.

[33] T. Jiang, C. Zhang, and Q.-M. Sun, "Green job shop scheduling problem with discrete whale optimization algorithm," *IEEE Access*, vol. 7, pp. 43153–43166, 2019.

[34] T. Jiang, C. Zhang, H. Zhu, J. Gu, and G. Deng, "Energy-efficient scheduling for a job shop using an improved whale optimization algorithm," *Mathematics*, vol. 6, no. 11, p. 220, Oct. 2018.

[35] M. Allaoui, B. Ahiod, and M. El Yafrani, "A hybrid crow search algorithm for solving the DNA fragment assembly problem," *Expert Syst. Appl.*, vol. 102, pp. 44–56, Jul. 2018.

[36] H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljarah, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita, "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowl.-Based Syst.*, vol. 154, pp. 43–67, Aug. 2018.

[37] A. B. Ali, G. Luque, E. Alba, and K. E. Melkemi, "An improved problem aware local search algorithm for the DNA fragment assembly problem," *Soft Comput.*, vol. 21, no. 7, pp. 1709–1720, Apr. 2017.

[38] E. Alba and G. Luque, "A new local search algorithm for the DNA fragment assembly problem," in *Proc. Eur. Conf. Evol. Comput. Combinat. Optim.* Berlin, Germany: Springer, 2007, pp. 1–12.

[39] L. Li and S. Khuri, "A comparison of DNA fragment assembly algorithms," in *Proc. METMBS*, vol. 4, 2004, pp. 329–335.

[40] X. Huang, "CAP3: A DNA sequence assembly program," *Genome Res.*, vol. 9, no. 9, pp. 868–877, Sep. 1999.

[41] H. Chen, W. Li, and X. Yang, "A whale optimization algorithm with chaos mechanism based on quasi-opposition for global optimization problems," *Expert Syst. Appl.*, vol. 158, Nov. 2020, Art. no. 113612.

[42] N. Neggaz, A. A. Ewees, M. A. Elaziz, and M. Mafarja, "Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection," *Expert Syst. Appl.*, vol. 145, May 2020, Art. no. 113103.

[43] E. Rodríguez-Esparza, L. A. Zanella-Calzada, D. Oliva, A. A. Heidari, D. Zaldivar, M. Pérez-Cisneros, and L. K. Foong, "An efficient harris hawks-inspired image segmentation method," *Expert Syst. Appl.*, vol. 155, Oct. 2020, Art. no. 113428.

[44] A. Zareie, A. Sheikhahmadi, and M. Jalili, "Identification of influential users in social network using gray wolf optimization algorithm," *Expert Syst. Appl.*, vol. 142, Mar. 2020, Art. no. 112971.

[45] M. Abdel-Basset, D. El-Shahat, I. El-Henawy, V. H. C. de Albuquerque, and S. Mirjalili, "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection," *Expert Syst. Appl.*, vol. 139, Jan. 2020, Art. no. 112824.

[46] S. Balochian and H. Baloochian, "Social mimic optimization algorithm and engineering applications," *Expert Syst. Appl.*, vol. 134, pp. 178–191, Nov. 2019.

[47] Y. Zhang, M. Ma, and Z. Jin, "Backtracking search algorithm with competitive learning for identification of unknown parameters of photovoltaic systems," *Expert Syst. Appl.*, vol. 160, Dec. 2020, Art. no. 113750.

[48] R. J. Parsons, S. Forrest, and C. Burks, "Genetic algorithms, operators, and DNA fragment assembly," *Mach. Learn.*, vol. 21, nos. 1–2, pp. 11–33, 1995.
[49] A. J. Nebro, G. Luque, F. Luna, and E. Alba, "DNA fragment assembly using a grid-based genetic algorithm," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2776–2790, Sep. 2008.
[50] J. A. Hughes, S. Houghten, and D. Ashlock, "Restarting and recentering genetic algorithm variations for DNA fragment assembly: The necessity of a multi-strategy approach," *Biosystems*, vol. 150, pp. 35–45, Dec. 2016.
[51] D. Bucur, "De novo DNA assembly with a genetic algorithm finds accurate genomes even with suboptimal fitness," in *Proc. Eur. Conf. Appl. Evol. Comput.* Cham, Switzerland: Springer, 2017, pp. 67–82.
[52] M. Rathee, K. Dilip, and R. Rathee, "DNA fragment assembly using quantum-inspired genetic algorithm," in *Exploring Critical Approaches of Evolutionary Computation*. Hershey, PA, USA: IGI Global, 2019, pp. 80–98.
[53] G. Minetti and E. Alba, "Metaheuristic assemblers of DNA strands: Noiseless and noisy cases," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1–8.
[54] I. Rajagopal and U. Sankareswaran, "An adaptive particle swarm optimization algorithm for solving DNA fragment assembly problem," *Current Bioinf.*, vol. 10, no. 1, pp. 97–105, Mar. 2015.
[55] K.-W. Huang, J.-L. Chen, C.-S. Yang, and C.-W. Tsai, "A memetic particle swarm optimization algorithm for solving the DNA fragment assembly problem," *Neural Comput. Appl.*, vol. 26, no. 3, pp. 495–506, Apr. 2015.
[56] G. M. Mallen-Fullerton and G. Fernandez-Anaya, "DNA fragment assembly using optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 1570–1577.
[57] K.-W. Huang, J.-L. Chen, and C.-S. Yang, "A hybrid PSO-based algorithm for solving DNA fragment assembly problem," in *Proc. 3rd Int. Conf. Innov. Bio-Inspired Comput. Appl.*, Sep. 2012, pp. 223–228.
[58] P. Vidal and A. Olivera, "Solving the DNA fragment assembly problem with a parallel discrete firefly algorithm implemented on GPU," *Comput. Sci. Inf. Syst.*, vol. 15, no. 2, pp. 273–293, 2018.
[59] A. B. Ezzeddine, S. Kasala, and P. Navrat, "Applying the firefly approach to the DNA fragments assembly problem," Annales Univ. Sci. Budapest., Sect. Comp., Tech. Rep., 2014, pp. 69–81. [Online]. Available: http://ac.inf.elte.hu/Vol_042_2014/069_42.pdf
[60] A. B. Ali, G. Luque, and E. Alba, "An efficient discrete PSO coupled with a fast local search heuristic for the DNA fragment assembly problem," *Inf. Sci.*, vol. 512, pp. 880–908, Feb. 2020.
[61] K.-W. Huang, J.-L. Chen, C.-S. Yang, and C.-W. Tsai, "A memetic gravitation search algorithm for solving DNA fragment assembly problems," *J. Intell. Fuzzy Syst.*, vol. 30, no. 4, pp. 2245–2255, Mar. 2016.
[62] E. D. Ulker, "Adaptation of harmony search algorithm for DNA fragment assembly problem," in *Proc. SAI Comput. Conf.*, Jul. 2016, pp. 135–138.
[63] R. Indumathy, S. U. Maheswari, and G. Subashini, "Nature-inspired novel cuckoo search algorithm for genome sequence assembly," *Sadhana*, vol. 40, no. 1, pp. 1–14, Feb. 2015.
[64] P. Meksangsouy and N. Chaiyaratana, "DNA fragment assembly using an ant colony system algorithm," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3, 2003, pp. 1756–1763.
[65] J. S. Firoz, M. S. Rahman, and T. K. Saha, "Bee algorithms for solving DNA fragment assembly problem with noisy and noiseless data," in *Proc. 14th Int. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2012, pp. 201–208.
[66] Z. Zhou, B. Yang, and W. Hou, "Association classification algorithm based on structure sequence in protein secondary structure prediction," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6381–6389, Sep. 2010.
[67] G. M. Mallén-Fullerton, J. A. Hughes, S. Houghten, and G. Fernández-Anaya, "Benchmark datasets for the DNA fragment assembly problem," *Int. J. Bio-Inspired Comput.*, vol. 5, no. 6, pp. 384–394, 2013.
[68] Y. Ling, Y. Zhou, and Q. Luo, "Lévy flight trajectory-based whale optimization algorithm for global optimization," *IEEE Access*, vol. 5, pp. 6168–6186, 2017.
[69] G. Kaur and S. Arora, "Chaotic whale optimization algorithm," *J. Comput. Des. Eng.*, vol. 5, no. 3, pp. 275–284, Jul. 2018.
[70] S. M. Bozorgi and S. Yazdani, "IWOA: An improved whale optimization algorithm for optimization problems," *J. Comput. Des. Eng.*, vol. 6, no. 3, pp. 243–259, Jul. 2019.
[71] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.
[72] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, pp. 106–130, Oct. 2000.
[73] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.
[74] W. Haynes, "Wilcoxon rank sum test," in *Encyclopedia of Systems Biology*. New York, NY, USA: Springer, 2013, pp. 2354–2355, doi: 10.1007/978-1-4419-9863-7.
[75] E. Alba, A. Nakib, and P. Siarry, *Metaheuristics for Dynamic Optimization*. Bristol, U.K.: IEEE, 2013.
[76] L. Qin, Q. Jiang, Z. Zou, and Y. Cao, "A queen-bee evolution based on genetic algorithm for economic power dispatch," in *Proc. 39th Int. Universities Power Eng. Conf. (UPEC)*, vol. 1, 2004, pp. 453–456.

MOHAMED ABDEL-BASSET (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in operations research from the Faculty of Computers and Informatics, Zagazig University, Egypt. He is currently an Associate Professor with the Faculty of Computers and Informatics, Zagazig University. He has published more than 200 papers in international journals and conference proceedings. His current research interests are optimization, operations research, data mining, computational intelligence, applied statistics, decision support systems, robust optimization, engineering optimization, multi-objective optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks. He is working on the application of multiobjective and robust metaheuristic optimization techniques. He is also an/a editor/reviewer in different international journals and conferences.

REDA MOHAMED received the B.Sc. degree from the Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Egypt. His research interests include robust optimization, multiobjective optimization, swarm intelligence, evolutionary algorithms, and artificial neural networks. He is working on the application of multiobjective and robust metaheuristic optimization techniques in computational intelligence.

KARAM M. SALLAM received the Ph.D. degree in computer science from the University of New South Wales at Canberra, Australian Force Academy, Canberra, Australia, in 2018. He is currently a Lecturer at Zagazig University, Zagazig, Egypt. His current research interests include evolutionary algorithms and optimization, constrained-handling techniques for evolutionary algorithms, operation research, machine learning, deep learning, cybersecurity, and the IoT. He was the winner of the IEEE-CEC2020 Competition. He serves as an organizing committee member for different conferences in the evolutionary computation field and a reviewer for several international journals.

RIPON K. CHAKRABORTTY (Member, IEEE) received the B.Sc. and M.Sc. degrees in industrial and production engineering from the Bangladesh University of Engineering and Technology, in 2013 and 2009, respectively, and the Ph.D. degree in computer science from the Bangladesh University of Engineering and Technology, in 2017. He is currently a Lecturer in system engineering and project management with the School of Engineering and Information Technology, The University of New South Wales (UNSW), Canberra, Australia. He has written two book chapters and over 50 technical journal and conference papers. His research interests include a wide range of topics in operations research, optimization problems, project management, supply chain management, and information systems management.

MICHAEL J. RYAN (Senior Member, IEEE) is currently the Director of the Capability Systems Centre, The University of New South Wales, Canberra. He lectures and regularly consults in a range of subjects, including communications systems, systems engineering, requirements engineering, and project management. He is the author/coauthor of twelve books, three book chapters, and over 250 technical articles and reports. He is a Fellow of the Engineers Australia, a Fellow of the International Council on Systems Engineering, and a Fellow of the Institute of Managers and Leaders. He is the Co-Chair of the Requirements Working Group in the International Council on Systems Engineering (INCOSE).