A Many-Objective Memetic Generalized Differential Evolution Algorithm for DNA Sequence Design

SHAH BANO, MARYAM BASHIR, AND IRFAN YOUNAS
FAST School of Computing, National University of Computer and Emerging Sciences, Lahore 54700, Pakistan
Corresponding author: Irfan Younas (irfan.younas@nu.edu.pk)

ABSTRACT Designing reliable sequences of DNA (Deoxyribonucleic Acid) is a critical task in the fields of DNA computing, and nanotechnology. The quality and reliability of the DNA sequence can directly affect the accuracy of the processing of information stored in sequences. This problem of designing reliable sequences belongs to the NP-hard class of problems. It has many incompatible design criteria, which cannot be optimized at the same time. Many objective evolutionary algorithms can balance conflicting design criteria by using a diverse population of solutions. This paper proposes an opposition-based Memetic Generalized Differential Evolution (MGDE3) to handle four conflicting design criteria for reliable DNA sequence design. Opposition-based learning and local search strategies are suggested to strengthen the exploitative and exploitative properties of the proposed MGDE3. The proposed algorithm is benchmarked with small, medium, and large data sets against 7 highly-cited many-objective and multi-objective algorithms. Experimental results and statistical analysis reveal that MGDE3 significantly outperforms the compared algorithms. The proposed method generates reliable real-life sequences of DNA that are substantially better than the DNA sequences generated by other considered algorithms.

INDEX TERMS Differential evolution, DNA sequence design, evolutionary algorithm, many-objective optimization.

I. INTRODUCTION The design of DNA sequence libraries is a critical task in many areas of nanotechnology and bioinformatics. The information in DNA is stored as four chemical bases C, A, G, and T [1]. Figure 1 (a) shows a nucleotide of DNA, which is made of phosphate, a pentose sugar, and an organic base. Figure 1 (b) shows a DNA sequence. It is a convention to represent start by 5’ (five prime) and end 3’ (three prime) for DNA sequence. There are four nucleotide bases of DNA, which are Adenine, Cytosine, Guanine, and Thymine. DNA libraries (collection of DNA fragments) can represent a huge amount of information. Due to this reason, DNA libraries are considered very useful for many areas of bioinformatics research. Elbaz et al. [1] used DNA sequence libraries to generate Nano-scale structures from nucleic acids and DNA sequences.

In past research [2], different design criteria, which are required for stable DNA sequences, have been proposed.

The associate editor coordinating the review of this manuscript and approving it for publication was Sun Junwei.
Traditional optimization methods (e.g., dynamic programming, and such other heuristic approaches) do not work for this problem because other reliable DNA sequence design involves many conflicting design criteria, which should avoid undesirable hybridizations. In the last decade, researchers have proposed many techniques to generate reliable DNA sequence libraries efficiently [3]. These techniques are based on a single design criterion by combining different design criteria into one optimization objective. These strategies consider only one part of the problem and due to this reason, the optimal solution is not obtained.

In the literature, multi-objective evolutionary algorithms (MOEAs) are widely used in tackling the DNA sequence design problem [4], [5] because they tend to optimize multiple objectives and consider conflicting design criteria. In this study, the problem of designing sequence of DNA is formulated as a many-objective optimization problem (MaOP). Previous MOEAs approaches combine different design criteria into mono optimization function and formulate a multi-objective optimization problem (MOP). MOEAs have shown promising results for the optimization problems having 2 or 3 objectives. The performance of existing MOEAs deteriorates when the number of objectives increase (due to an enormous number of non-dominated solutions). This design problem is considered a MaOP due to its four design criteria (Hairpin, Similarity, Continuity, and H-measure). The opposing nature of objectives does not allow us to optimize them simultaneously. The conflicting nature of the 4 objectives and the constraints make this problem hard. To solve a problem optimally, there is a need to maintain a balance between the exploratory and exploitative capability of the algorithm. Due to the complex nature of sequence design of DNA problem, there is a need to design an approach, which can appropriately balance the exploitation and exploration to solve the problem effectively. In this research, a Memetic Generalized Differential Evolution algorithm (MGDE3) is suggested to tackle the problem of designing sequence of DNA. MGDE3 algorithm uses operators based on the concept of differential vectors, where the newly updated position of a solution is based on the difference of positions of solutions. In early iterations of the algorithm, the generated solutions are widely distributed throughout the search space of the problem, therefore large differences in position vectors (solutions) help to investigate different parts of the search space. However, during the later phase of iterations, the difference in vectors becomes smaller, which helps to exploit the promising regions of the search space. Moreover, to enhance the diversity of the current population, we generate an opposite solution of an offspring with some probability using opposition-based learning. To enhance the exploitative capability of GDE3, we incorporate a local search strategy, which improves the updated solution at hand by searching and replacing it with a better solution in the neighborhood. We have done experiments using DNA sequence data sets of different size and compared the results of our proposed MGDE3 with well cited multi and many-objective evolutionary algorithms (MaOEAs) like NSGA-III [6], NSGA-II [7], MOEA/D [8], DBEA [9], GDE3 [10], RVEA [11], and MaOEAC [12]. The results show that MGDE3 has better performance than the compared methods. The main contributions of this study are as follows:

- In this paper, we formulate the problem of designing DNA sequence as a MaOP and propose an opposition-based Memetic Generalized Differential Evolution to find optimal trade-off solutions of the problem effectively.

- The proposed metaheuristic combines opposition-based learning, local search, and the differential evolution based strategy in a synergetic way to exploit the benefits of each approach. The proposed opposition-based learning and the local search strategy help to keep a balance between the explorative and exploitative behavior of the proposed method. Opposition-based learning is incorporated to enhance the explorative property of the algorithm, which aids to prevent the premature convergence of the presented approach. Moreover, to exploit the promising parts of the search space, we suggest a local search strategy, which results in fast convergence.

- The proposed opposition-based MGDE3 is evaluated on 4 datasets of different sizes against 7 widely used state-of-the-art MOEAs and MaOEAs. The experimental results and the significance test reveal that the proposed approach outperforms all the compared metaheuristics. The excellent performance on datasets of different sizes against widely used metaheuristics shows that the proposed many-objective Memetic GDE3 is an effective method for generating reliable real-world DNA sequences.

The rest of the sections are arranged as follows. Section II presents the literature review on many and multi-objective optimization methods and the techniques used for designing sequence of DNA. Section III describes the problem formulation and Section V presents the proposed opposition-based MGDE3. The experimental results and comparative analysis are shown in Section VI. In the end, Section VII concludes the study with some future guidelines.

II. RELATED WORK

In this section, we present the literature review related to optimization algorithms for many and multi-objective optimization. We also review optimization techniques applied in past research for the problem of designing the sequence of DNA.

A. MULTI-OBJECTIVE AND MANY-OBJECTIVE OPTIMIZATION

A problem with 2 or 3 objectives is called MOP and if the number of objectives is more than 3 then it is called MaOP. Multi-objective algorithms have been successful when there are 2 or 3 objectives; however, as the number of objectives increase, the algorithms perform poorly. To conquer these problems, several MaOEAs have been
proposed [6], [13]–[15]. MaOEAs can be classified into several categories. We briefly review the algorithms from each class in the following.

Relax dominance-based algorithms change the dominance criteria of solutions by easing dominance conditions. There are two approaches to modifying dominance. Some examples of this approach are algorithms like \( \epsilon \)-MOEA [14], \( \epsilon \)-NSGA-II [16], MDMOEAs [17] and LD [18]. Diversity based algorithms mitigate the unfavorable effects of diversity preservation. Some many-objective algorithms using a diversity-based approach are region search evolutionary algorithm [19], and NSGA-II-SDE [14]. Aggregation based algorithms rank solutions by aggregating individual information or pairwise information. Some of the value-based aggregation algorithms are MSOPS [20], MOEA/D [21], surrogate assisted decomposition-based evolutionary algorithm [22], and decomposition-based many-objective algorithm [23]. The indicator-based approach uses solution set indicators for the direction. These indicators are metrics for determining the quality of the solution set. Some examples of these indicators are hypervolume (SMS-EMOA [24]), the distance-based indicator (DDE [25], AGE-II [24], hypervolume based algorithm [26], and MOMBI [27]) algorithms. These methods are computationally expensive so these have not been widely used. There is a class of optimizers, which use a reference set to evaluate the quality of the candidate solutions. There are many variants, depending on how the reference set is managed and how it evaluates the candidate solutions. The most popular reference-based approach is NSGA-III [6], others include many-objective Bat algorithm [28], KnRVEA [29], many objective cuckoo [30], and two-archive strategy based algorithm [31]. The preference-based methods incorporate user preference to direct the search towards a specific area of the search space. This preference can be specified before the optimization, during optimization, or after optimization. Some examples of these three categories of preference-based algorithms are a priori (\( \epsilon \)-Constraint+CDE [32], 2p-NSGA-II [33], MQEA-PS2 [34]), interactive (P-MOET [35]), and posteriori methods. Dimensionality reduction methods help in reducing the objectives count and modify a many-objective problem with many objectives into a problem with fewer objectives. These methods are computationally cheaper but some information might be lost due to fewer objectives. Some examples are L-PCA [36], Exact-MOSS/k-EMOSS [37], and OC-ORA [38].

In the literature, different multi and many-objective optimizers have been developed to tackle different optimization problems effectively. However, no optimization algorithm can perform best for all optimization problems. Different categories of algorithms have different limitations. For instance, dominance-based algorithms, which use dominance depth as the fitness scheme, are good in convergence but may lack diversity. Aggregation based algorithms use different approaches to combine the objectives, which make them ineffective in handling irregular Pareto fronts. Similarly, the indicator-based approaches are computationally expensive, which makes them infeasible for most of the optimization problems.

B. OPTIMIZATION TECHNIQUES FOR SEQUENCE DESIGN OF DNA PROBLEM

In 1994, Adleman [39] proposed the first computational model of DNA computing and after that, various techniques have been developed for efficiently generating reliable DNA sequences. Some systems have been designed using DNA strand displacement [40] but our focus is on algorithms designed for sequence design of DNA. Many algorithms in the past have been proposed by using random searches for the generation of DNA sequences. Penchovsky [41] used a random search algorithm. These techniques are not efficient, as they require enormous computational resources [41]. Shin et al. [42] formulated this problem as a multi-objective problem for the very first time, using six different bio-chemical design criteria, and tackle it using an evolutionary algorithm.

An improved non-dominated sorting genetic algorithm II (NSGA-II) [43] was proposed for this problem by using five different constraints. González et al. [6] has applied NSGA-II with four design criteria to design a DNA sequence library. Salido et al. [44] modified the NSGA-II algorithm by introducing the new representation of chromosome and recombination techniques.

Gonzalez and Vega-Rodriguez [45] proposed a multi-objective version of the Firefly algorithm (MOA) for designing DNA sequences. The general scheme of this algorithm is like the firefly algorithm but the author has adopted the non-dominating sorting technique and Pareto dominance technique from NSGA-II [7]. In the proposed technique, to maintain the updated record of the best solution the non-dominated solution archive (NDS archive) the technique of PAES (Pareto Archived Evolution Strategy) [46] is used.

Gonzalez et al. [47] proposed multi-objective Differential Evolution (DE) algorithm with Pareto tournaments for the problem of sequence design of DNA. They used crowding distance and sorting using the dominance method of NSGA-II [7]. Moreover, they used the NDS archive technique of PAES [46]. Gonzalez et al. [48] developed MOABC (multi-objective artificial bee colony) algorithm for the same problem. This algorithm is similar to the original ABC (artificial bee colony) algorithm, but the author has adopted features like a non-dominating sorting technique from NSGA-II [7] and the NDS archive technique of PAES [46]. Gonzalez et al. [49] designed a hybrid and multi-Objective version of Teaching learning based optimizer.

The existing work on designing a sequence of DNA formulates this problem as a MOP and solves it using different MOEAs. Multi-objective metaheuristics have shown promising results for tackling the optimization problems with less than 3 objectives. However, as the objectives increase, the performance of existing multi-objective algorithms deteriorates because of an increase in non-dominated solutions, which
makes the dominance criteria of selection ineffective [50]. Due to more than 3 objectives, this paper is the first attempt to formulate this problem as a MaOP. In this study, an effective MaOEA is proposed to generate better and reliable real-world DNA sequences. The proposed memetic GDE3 algorithm combines opposition-based learning, local search, and the differential evolution based strategy to get the benefits of each approach. Memetic algorithms [51] and differential evolution algorithms [52], [53] have been used in past studies for other problems. This study is the first attempt to use a memetic generalized differential evolution algorithm for this problem. Opposition-based learning is used to enhance the explorative property of the algorithm, which aids to prevent the premature convergence of the proposed algorithm. Moreover, to exploit the promising parts of the search space, we suggest a local search strategy, which results in fast convergence.

III. THE PROBLEM OF DESIGNING DNA SEQUENCE

In this section, the general problem statement of the problem is presented with its objectives and constraints.

A. GENERAL PROBLEM STATEMENT

DNA library comprises a set of DNA sequences, in which each sequence consists of m nucleotides. DNA sequences are a series of letters (C, A, G, and T). The nucleotides in molecules of DNA are arranged using these letters. The generation of reliable DNA sequence libraries is known as a problem. A biochemical reaction between strands of DNA is considered a computational process in the field of DNA computing. The reliability of DNA sequences depends on whether they interact with compatible sequences. Garzon and Deaton [54] introduced the concept of Watson Crick base-pairing (WCBP). WCBP hybridizes the sequence of DNA and the sequence of complement base-pairing. The process of hybridization helps in manipulating the information stored in DNA sequences. In Watson Crick base pairs, it is ensured that Thymine (T) combines with Adenine (A) and Cytosine (C) combines with Guanine (G). This combination of base pairs allows maintenance of the regular helical structure of DNA and make the DNA strand more stable and reliable. Figure 2 shows the example of Watson crick pairing. In this example, two DNA strands are pairing in the opposite direction. Figure 2 shows hydrogen bonds between corresponding base pairs (A is linked to T and G is linked to C). The most crucial step is a hybridization between the DNA sequence and its complement base-pairing sequence. DNA sequences should avoid undesirable hybridization because it causes errors during biological reactions among sequences. Undesirable hybridization should be controlled at the design phase. Thus, for this purpose, such design criteria should be selected which improves the reliability and stability of DNA sequences. These biological design criteria are of four types [2], [55], and are explained in the next section.

B. MANY-OBJECTIVE PROBLEM (MaOP) FORMULATION

The problem of designing sequences of DNA can be constructed as a MaOP as follows: Considering this problem, we have a DNA library $D_l$ that consists of $l$ DNA sequence. Each sequence is composed of $m$ number of nucleotides and supposes DNA library $D_l$ has two DNA sequences $L_i$ and $L_j$, $m$ is the length of DNA sequences $L_i$ and $L_j$. Every element of these sequences belongs to the set $N$. DNA nucleotides and a gap space, denoted by ‘–’ are contained in set $N$. The optimization problem consists of minimizing four objectives: similarity ($f_1$), H-measure ($f_2$), continuity ($f_3$), and hairpin ($f_4$) subject to the two problem constraints, which are melting temperature ($c_1$) and GC ratio ($c_2$). DNA nucleotide bases are encoded with numerical values, 0, 1, 2, 3 encode A, T, G, and C respectively. For instance sequence “ATGAC” is represented as “01203”.

1) SIMILARITY OBJECTIVE

In this objective, the count of sub-sequences, which are similar, is minimized. The similarity between sequences of every pair in the DNA library is computed in the same direction. This process also includes position gaps and shifts. The similarity of parallel DNA sequences consists of two similarity criteria: (i) SimDis (discontinuous similarity), and (ii) SimCont (the common subsequence, which is the largest). This similarity computation is illustrated in Figure 3. Continuous similarity computes the same substring among two sequences and discontinuous similarity computes overall similarity. The similarity computation without any shift, and with the shift by base 1 are represented in Figure 3 (a), and Figure 3 (b), respectively.

$$F_{Similarity}(D_l) = \sum_{j=1}^{l} \sum_{k=1}^{l} Similarity(L_j, L_k),$$

where $l$ represents the number of sequences in $D_l$. $L_j$ and $L_k$ are two sequences in library $L$.

$$Similarity(r, s) = Max_g st(SimDis(r, FShift(s(−)g s, st)) + SimCont(r, FShift(s(−)g s, st))),$$

where $r$ and $s$ are two DNA sequences. The copy of sequence $s$ is generated and gaps (‘−’) are inserted between sequence $s$ and the copy of $s$, where $g$ is

$$0 \leq g \leq n − 1,$$
here \( n \) represents the count for nucleotide bases in \( r \) and \( s \). The function \( F_{\text{Shift}} \) is used to shift the sequence \( s \) by \( st \) bases, here \(|st| \leq n\).

\[
\text{SimDis}(r, s) = Th\left(\sum_{i=0}^{n} eqi(r_i, s_i), Sd|n|\right),
\]

and the copy of \( s \), where \( g \) is \( 0 \leq g \leq n1 \), and \( n \) represents the total count of the nucleotide bases for \( r \) and \( s \). The sequence \( s \), is shifted with function \( F_{\text{Shift}} \) by \( st \) bases, here \(|st| \leq n\).

\[
HmDis(r, s) = Th\left(\sum_{i=0}^{k} cb(r_i, s_i), Hd|s|\right),
\]

and \( Hd \) is a real value between 0 and 1; \( Hc \) is an integer between \( n \) and 1, in which \( n \) represents the count of bases for \( s \) and \( r \). The value of function \( Th(t, t_{\text{value}}) \) is the threshold function as defined in equation 5.

\[
cb(m, n) = \begin{cases} 
1, & \text{if } m = n' \\
0, & \text{otherwise,}
\end{cases}
\]

In the above equation, \( n' \) represent Watson crick complement of nucleotide base \( n \) and \( \text{subcb}(r, s, i) \) is the length of DNA pairs.

2) H-MEASURE OBJECTIVE

In this objective, the hamming distance of each DNA sequence with other DNA sequences in opposite direction, from the DNA library is calculated. This prevents the undesirable chemical reaction between sequences.

\[
H\text{measure}(Di) = \sum_{j=1}^{l} \sum_{k=1}^{l} H\text{measure}(L_j, L_k),
\]

where \( l \) represents the count of sequences in \( Di \). \( L_j \) and \( L_k \) are two sequences in library \( L \). H-measure of parallel DNA sequences consist of two types of similarity: (i) \( H\text{mDis} \) (discontinuous similarity), and (ii) \( H\text{mCont} \) (the common subsequence, which is the largest and continuous). Figure 3 (c) and (d) show \( H \) measure computation. Figure 3 (c) shows \( H \) measure computation without shift and Figure 3 (d) shows \( H \) measure computation with the shift by base 2. It is the convention to represent the first sequence from start 5’ to end 3’ and the second sequence from end 3’ to start 5’.

\[
H\text{measure}(r, s) = \text{Max}_{g,s}(H\text{mDis}(r, F_{\text{Shift}}(s(-)^g s, st)),
+ H\text{mCont}(r, F_{\text{Shift}}(s(-)^g s, st))),
\]

where \( r \) and \( s \) are two DNA sequences. The copy of sequence \( s \) is generated and \( g \) are inserted between sequence \( s \) and the copy of \( s \), where \( g \) is \( 0 \leq g \leq n1 \), and \( n \) represents the total count of the nucleotide bases for \( r \) and \( s \). The sequence \( s \), is shifted with function \( F_{\text{Shift}} \) by \( st \) bases, here \(|st| \leq n\).

\[
HmDis(r, s) = Th\left(\sum_{i=0}^{k} cb(r_i, s_i), Hd|s|\right),
\]

\[
HmCont(r, s) = \sum_{i=0}^{k} Th(\text{subseq}(r, s, i), Hc),
\]

3) HAIRPIN OBJECTIVE

The hairpin penalty helps to prevent DNA sequences from generating secondary structures. It is determined at every position of DNA sequence by considering a minimum number of hybridized pairs \( S_{\text{min}} \) and a minimum length of hairpin loop \( V_{\text{min}} \).

\[
F_{\text{hairpin}}(Di) = \sum_{k=1}^{l} \text{Hairpin}(L_k),
\]

\[
\text{Hairpin}(m) = \sum_{k=1}^{l} \sum_{v=V_{\text{min}}}^{V_{\text{min}}} \sum_{j=1}^{plq(s, v, k)} Th\left(\sum_{j=1}^{plq(s, v, k)} cb(m_{(s+k+j)}, m_{(s+k+v+j)}), plq(s, v, k)/2\right),
\]

where \( S_{\text{min}} \) represents the minimum number of hybridized pairs, \( V_{\text{min}} \) represents minimum hairpin loop length and \( plq(s, v, k) = \text{min}(s+k, l-v-k-s) \) represents the count of base pairs at the center \( s+k+v/2 \) of hairpin. \( Th(t, t_{\text{value}}) \) is the threshold function as defined in equation 5.
4) CONTINUITY OBJECTIVE

The continuity objective is used to prevent the consecutive repetition of the same base of the sequence. A continuity threshold is used for penalizing consecutive occurrences of the same base in a sequence. If the continuity value exceeds the threshold, it will make the DNA structure unstable.

\[ F_{\text{continuity}}(Dl) = \sum_{i=1}^{l} \mathit{continuity}(L_i), \] (13)

where \( n \) is total number of sequences in library \( Dl \) and \( \mathit{continuity}(L_i) \) is defined as follows:

\[ \mathit{continuity}(m) = \sum_{i=1}^{n-C\mathit{Thresh}} \mathit{Th}(\mathit{con}_c(m, i), C\mathit{Thresh}), \] (14)

where \( m \) is the sequence, \( n \) is the total number of bases in sequence \( m \), \( C\mathit{Thresh} \) is continuity threshold, and \( \mathit{Th}(a, Th) \) is the threshold function as defined in equation 5. \( \mathit{con}_c(m, i) \) is a continuity function, with value \( c \) if a base (one of the four bases A, C, T, G) has at most \( c \) consecutive occurrences in the sequence. This function is applied for every base and the maximum value of all bases is returned.

5) GC RATIO CONSTRAINT

GC content of DNA affects the stability of DNA sequences. It is measured by GC ratio, which is computed as the percentage of cytosine (C) and guanine (G) in generated DNA sequences.

\[ \mathit{GC\mathit{content}}(m) = (NG(m) + NC(m))/n \times 100, \] (15)

Here \( n \) represents the count of bases in the DNA sequence. The concentration of cytosine and guanine are represented by \( NC \) and \( NG \), respectively.

6) MELTING TEMPERATURE CONSTRAINT

Stable DNA structures perform more reliable operations as compared to unstable DNA structures. The base composition of DNA is used for determining DNA stability. The nearest neighbor technique [56] is the most commonly used method for calculating this feature.

\[ \mathit{Melting\mathit{Temp}}(m) = \frac{H^0(x)}{S^0(x) + R.\ln \left( \frac{C_T}{T} \right)}, \] (16)

where \( m \) represents a sequence, \( R \) is a constant with a value of 1.987 cal/Kmol, and \( C_T \) is the concentration of the sequence of DNA, its value is equal to 10nM [56]. \( H^0 \) and \( S^0 \) donate predicted enthalpies and entropies respectively.

C. REPRESENTATION OF THE INDIVIDUALS

It will be helpful to illustrate how individuals are represented in evolutionary algorithms. Representation of individuals is important for understanding how genetic algorithms solve the problem. Each individual is made of \( m \) DNA sequences where each DNA sequence has \( n \) bases. The values of four design criteria and 2 constraints are also stored with each DNA sequence along with the average values of four design criteria.

IV. GENERALIZED DIFFERENTIAL EVOLUTION ALGORITHM (GDE3)

A. INSPIRATION

GDE3 [10], is based on the Generalized Differential Evolution (GDE) algorithm. Differential Evolution (DE) (inspired by the genetic algorithm) is a population-based evolutionary algorithm. In DE, initially, the random population will be generated, then at each generation, it will be improved with the help of DE mutation and crossover operators. Selection is made at random from the population after initialization. The fitness of the population is evaluated after applying DE operators. The optimal parent and offspring are selected for the next generation. GDE3 [10] replaces the genetic algorithm portion of NSGA-II [6] with selection, mutation, and recombination of DE.

B. LOCAL SEARCH

Local search is a heuristic, which helps to solve complex optimization problems. The local search strategy has the ability to exploit information about the search space in generating new neighborhoods. It enhances the exploitative capability of the algorithm, which helps in improving the solutions by exploiting the information from the solutions at hand. The local search strategies proposed in this paper can meet the above demands.

The local search keeps a single current state (candidate solution) and moves to neighboring states to improve it. Local search finds the best neighbor state according to some quality function. Algorithm 1 shows the general strategy of the local search when used for candidate solution \( y \). \( N(y) \) represents the set of neighbors of \( y \), and \( f(y) \) represents fitness or quality function of solution \( y \).

```
Algorithm 1 Local Search (y) with depth \( \delta \)
for \( \delta \) iterations do
    if \( \exists z \in N(y) \) with \( f(z) > f(y) \) then
        \( y = z \)
    end if
else
    stop with output \( y \)
end if
```

C. LOCAL SEARCH ENHANCED (Memetic) GDE3 ALGORITHM

In this paper, we propose a local search (LS) enhanced GDE3 algorithm for solving the problem of designing DNA sequence. To solve a problem optimally, there is a need to maintain a balance between the exploitative and explorative capability of the algorithm. The operators of the differential evolution algorithms are based on the concept of differential vectors.

During evolution, the newly updated position of a solution is based on the difference of positions of solutions. In early
Algorithm 2 MemeticGDE3: Local Search Enhanced (Memetic) GDE3 Algorithm Inspired From [10], [57]

INPUT: N, max\textsubscript{it}, x\textsubscript{lb}, x\textsubscript{ub}, LS\textsubscript{strategy}, F ∈ (F\textsubscript{min}, F\textsubscript{max})

t = 0;

P\textsubscript{t} = Initialize population uniform randomly;

Evaluate P\textsubscript{t} (Compute all objective function values of the population)

Main loop
while t < max\textsubscript{it} do

Q\textsubscript{t} = EvolveSolutions(N, P\textsubscript{t}, x\textsubscript{lb}, x\textsubscript{ub}, LS\textsubscript{strategy}, F)

Evaluate Q\textsubscript{t} (Compute all objective function values of the population)

R\textsubscript{t} = Q\textsubscript{t} ∪ P\textsubscript{t}

F = NonDominatedSorting(R\textsubscript{t})

P\textsubscript{t+1} = ∅
i = 1

until P\textsubscript{t+1} + F\textsubscript{i} ≤ N

CrowdingDistance(F\textsubscript{i})

P\textsubscript{t+1} = P\textsubscript{t+1} ∪ F\textsubscript{i}

i = i+1

sort in decreasing order on the basis of crowding distance

P\textsubscript{t+1} = P\textsubscript{t+1} ∪ F\textsubscript{i}[1 : (N - |P\textsubscript{t+1}|)]

\textbf{end while}

A neighbor is added to the neighbors’ population, represented by a variable name neighborsPopulation, if it dominates U. A solution Neighbor\textsubscript{i} dominates solution U if Neighbor\textsubscript{i} is no worse than U in all objectives and strictly better than U in at least one of the objectives. The last for loop finds one neighbor who is best among all generated neighbors in neighborsPopulation and replaces U with that neighbor.

2) LOCAL SEARCH STRATEGY 2
Algorithm 8 presents LS strategy 2. The neighbors are generated in the same way as in LS strategy 1 but this is a probabilistic step and it is performed if the fitness criteria mentioned in Algorithm 8 are satisfied. A neighbor is added to the neighbors’ population, represented by the variable name neighborsPopulation. After generating the neighbor solutions of U, we iterate over neighbors (i = 1 to size(neighborsPopulation)), evaluate the neighbor N\textsubscript{i}, and replace the solution U with N\textsubscript{i} if the N\textsubscript{i} dominates U. This loop is terminated when we find a solution N\textsubscript{i}, which dominates U.

3) LOCAL SEARCH STRATEGY 3
Algorithm 9 presents LS strategy 3. In this strategy, LS is applied with some probability. The input is a child solution U and its neighbors are generated by overwriting a random number from 0 to 3 at some index of U. In this strategy, we stop our search when we find a first immediate non-dominated neighbor and do not explore further. This strategy returns the first non-dominating neighbor it finds.

4) LOCAL SEARCH STRATEGY 4
Algorithm 10 presents LS strategy 4. This strategy is very similar to LS strategy 1. The neighbors are generated in the same way as in local search strategy 1, but this step is performed with some probability. The last for loop finds one neighbor who is best among all generated neighbors in neighborsPopulation and replaces U with that neighbor.

D. TIME COMPLEXITY OF MEMETIC GDE3 ALGORITHM
The memetic GDE3 algorithm uses crowding distance calculation and non-dominated sorting. Non-dominated sorting can be implemented in O(P\textsubscript{log\textsuperscript{K-1}P}) time [59]. Crowding distance is implemented in O(KP\textsubscript{log\textsuperscript{P}}) time [59].
algorithms where the optimization problem is divided into $n$ single-objective and multi-objective subproblems. There are many approaches to this decomposition step, the most popular approach is the weighted Tchebycheff approach. These subproblems are solved at the same time and knowledge from neighborhood problems is used for resolving a subproblem. MOEA/D provides computational efficiency, scalability with the number of objectives, and best performance for combinatorial optimization problems.

V. MULTI AND MANY OBJECTIVE EVOLUTIONARY ALGORITHMS FOR COMPARISON

This section describes many-objective optimization algorithms used for comparison to our proposed algorithm.

A. MOEA/D

MOEA/D [8], [21] is a many-objective optimization algorithm where the optimization problem is divided into a non-diminishing population size of $P$, and $max_t$ represent the number of iterations of the main loop respectively.

\begin{algorithm}
\caption{EvolveSolutions: Iterative Algorithm to Evolve Solutions for MemeticGDE3}
\begin{algorithmic}
\STATE {INPUT: $N$, $P$, $x_{lb}$, $x_{ub}$, $LS_{strategy}$, $F \in (F_{min}, F_{max})$}
\FOR{$p = 1, p \leq N, p++$}
\STATE Selection $X_1, X_2, X_3 \in \{1, 2, \ldots, P\}$
\STATE $j_{rand} \in 1, 2, \ldots, D$
\STATE $V = X_1 + F \ast (X_2, X_3)$, Creating mutant vector $V$
\STATE $x_{lb} \leq V(i) \leq x_{ub}$, ensure $V$ is within bounds
\STATE Mutation and Crossover:
\STATE $U(j) = \begin{cases} V(j), & \text{if } (rand(0, 1) < Cr) \text{OR}(j == j_{rand}) \\ X(i), & \text{otherwise} \end{cases}$
\IF{$LS_{strategy} == 2$}
\STATE \textbf{Opposition based learning on $U$}
\STATE $X_i = \text{ComputeOpponent}(U)$
\STATE Selection $X_i = \begin{cases} U, & \text{if } U \text{ dominates } X_i \\ X_i, & \text{otherwise} \end{cases}$
\STATE childrenList = childrenList $\cup$ $X_i$
\ELSE
\STATE childrenList = childrenList $\cup$ $U(j)$
\ENDIF
\ENDFOR
\STATE \textbf{Local Search on children}
\FOR{$p = 1, p \leq \text{childrenList}, p++$}
\STATE $U_p = \text{childrenList}(p)$
\IF{$LS_{strategy} == 1$}
\STATE childrenList(p) = LocalSearch1($U_p$)
\ENDIF
\IF{$LS_{strategy} == 2$}
\STATE childrenList(p) = LocalSearch2($U_p$)
\ENDIF
\IF{$LS_{strategy} == 3$}
\STATE childrenList(p) = LocalSearch3($U_p$)
\ENDIF
\IF{$LS_{strategy} == 4$}
\STATE childrenList(p) = LocalSearch4($U_p$)
\ENDIF
\ENDFOR
\STATE $P_t = P_t \cup \text{childrenList}$
\RETURN $P_t$
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{Non-dominatedSorting(P)}
\begin{algorithmic}
\FOR{$p \in P$}
\STATE $S_p = \emptyset$
\STATE $n_p = 0$
\FOR{$q \in P$}
\IF{$p \text{ dominates } q$}
\STATE $S_p = S_p \cup q$
\ENDIF
\ELSIF{$q \text{ dominates } p$}
\STATE $n_p = n_p + 1$
\ENDIF
\ENDFOR
\STATE $F_1 = F_1 \cup p$
\ENDFOR
\STATE $Q = \emptyset$
\FOR{$p \in F_1$}
\FOR{$q \in S_p$}
\IF{$n_q = n_q - 1$}
\STATE $q_{rank} = q_{rank} + 1$
\ENDIF
\ENDFOR
\STATE $Q = Q \cup q$
\ENDFOR
\STATE $i = i + 1$
\STATE $F_i = Q$
\WHILE{$F_i \neq \emptyset$}
\STATE $Q = \emptyset$
\FOR{$p \in F_i$}
\FOR{$q \in S_p$}
\IF{$n_q = n_q - 1$}
\STATE $q_{rank} = q_{rank} + 1$
\ENDIF
\ENDFOR
\STATE $Q = Q \cup q$
\ENDFOR
\ENDWHILE
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{CrowdingDistance(P)}
\begin{algorithmic}
\STATE $len = |P|$
\FOR{$j \in P$}
\STATE set $P[j]_{CD} = 0$
\ENDFOR
\FOR{$j = 2$ to $len - 1$}
\STATE $P[j]_{CD} = P[j - 1].o - P[j + 1].o + (F_{max} - F_{min})$
\ENDFOR
\end{algorithmic}
\end{algorithm}
Algorithm 6 ComputeOpponent: Algorithm for Finding an Opponent of a Child Solution [58]

Input: U
Generate random number between 0 and 1
if random number is less than 0.25 then
    for i ← 0 to size(U) do
        Let \( u_i \in [a_i, b_i] \)
        \( \bar{u}_i = a_i + b_i - u_i \)
        Generate random number \( rand_i \) between 0 and 1
        \( \bar{u}_i = u_i + rand_i \times (\bar{u}_i - u_i) = u_i + rand_i \times (a_i + b_i - 2u_i) \)
        \( u_i = \bar{u}_i \)
    end for
end if
return U

Algorithm 7 LocalSearch1

Input: U
for j ← 0 to size(U) do
    Neighbor\(_j\) ← generate a random value and overwrite this random value at index \( j \) of the child U
    Evaluate Neighbor\(_j\)
    if Neighbor\(_j\) dominates solution U then
        NeighborsPopulation ← Neighbor\(_j\)
    end if
end for
for i ← 0 to size(NeighborsPopulation) do
    U ← Find one neighbor from this population which optimizes all four objectives for two constraints
end for
return U

Algorithm 8 LocalSearch2

Input: U
for i ← 0 to size(all feasible children) do
    Sum1 ← Sum1 + sum of objective 1 and 3 for child \( i \)
    Sum2 ← Sum2 + sum of objective 2 and 4 for child \( i \)
end for
Mean1 ← Sum1/size(all feasible children)
Mean2 ← Sum2/size(all feasible children)
deltaFitness ← Mean2/100
Fitness1 ← sum of objective 1 and 3 for U
Fitness2 ← sum of objective 2 and 4 for U
Generate random number \( rand \) between 0 and 1
if Fitness1 < Mean1 - 0.1 AND Fitness2 < Mean2 - deltaFitness AND \( rand < 0.015 \) then
    for j ← 0 to size(U) do
        Neighbor\(_j\) ← generate random value and overwrite this random value at index \( j \) of child U
        Evaluate Neighbor\(_j\)
        if Neighbor\(_j\) dominates solution U then
            U = Neighbor\(_j\)
            break;
        end if
    end for
end if
return U

Algorithm 9 LocalSearch3

[!htb] Input: U
Produce a random number in the range of 0 and 1
if random number is less than a specified Probability then
    for j ← 0 to size(U) do
        Neighbor\(_j\) ← generate random value and overwrite this random value at index \( j \) of the child U
        Evaluate Neighbor\(_j\)
        if Neighbor\(_j\) has minimum value in all objectives subject to the constraints then
            return Neighbor\(_j\)
        end if
    end for
end if
return U

B. NSGA-III

Deb et al. proposed NSGA-III [6] many-objective algorithm. This algorithm is based on the NSGA-II [7] algorithm but some changes are made in its selection technique. NSGA-III uses a reference points based approach. This technique maintains population diversity and convergence. In each generation, the solutions are normalized and associated with a reference point. Selection based on the niching approach is used to create a next-generation population.

C. DBEA

DBEA [9] is an evolutionary algorithm based on decomposition. There are three phases in DBEA: initialization, recombination, and replacement.
Algorithm 10 LocalSearch4

Input: U
Produce a random number in the range of 0 and 1
if random number is less than a specified Probability then
    for j ← 0 to size(U) do
        Neighbor j ← generate a random value and overwrite this random value at index j of child
evaluate Neighbor j
        if Neighbor j dominates solution U then
            NeighborsPopulation ← Neighbor j
        end if
    end for
    for i ← 0 to size(NeighborsPopulation) do
        U ← Find one neighbor from this population which optimizes all four objectives for two constraints
    end for
end if
return U

1) In the initialization phase, K uniformly distributed reference vectors are generated by using the technique of Normal Boundary Intersection (NBI) [60]. The fitness value of each objective is calculated and the vector with the minimum value for all objectives is selected as the ideal point. The population is generated by using the technique of Latin Hyper-cube Sampling (LHS) A set of reference vectors is used for forming the Pareto front.

2) In the recombination phase, a new child is generated by mating each individual from the population with the randomly selected individual from the population by using SBX crossover. It helps in avoiding mating within the neighborhood. Each child is evaluated and the ideal point is computed. All generated children are scaled using the ideal point.

3) In the replacement phase, two distances \(d_1\) and \(d_2\) are computed for the child population (newly generated) and parent population (old population). Each newly generated child is compared with old parent population members and if all members in the parent population are feasible then the child replaces dominated individuals from the parent population. If all solutions in the parent population are infeasible then the child with optimal fitness value replaces the worst individual from the parent population. If all children fail to dominate any member from the parent population, then the replacement will occur based on a smaller \(d_2\) distance [9].

\[\text{D. RVEA}\]

RVEA [11] is a many-objective evolutionary algorithm based on the reference vector approach. This algorithm is inspired by MOEA/D [8], a decomposition-based approach. The search space is divided into subspaces by using reference vectors. The distance of solutions to reference vectors is calculated using an angle penalized distance approach. This algorithm promotes a diversity of solutions by generating solutions that cover a large area of the Pareto front.

\[\text{E. MaOEA/AC}\]

MaOEA/AC [12] is a many-objective evolutionary algorithm that uses an adaptive clustering-based selection strategy. The population of solutions is divided into N clusters. One solution is selected from each cluster to generate a varying set of solutions that help in exploring different areas of the search space.

\[\text{VI. EXPERIMENTS AND RESULTS}\]

In this section, the experimental methodology and results of experiments are presented. We have implemented and compared several many-objective optimization algorithms, including MOEA/D, NSGA-III, DBEA, GDE3, RVEA, MaOEA/AC, and our proposed Memetic GDE3 for the problem of designing sequence of DNA. According to statistical analysis, our proposed Memetic GDE3 performed better than other algorithms.

\[\text{A. DATA SETS}\]

We have used 4 different data sets of different sizes. The data sets are represented by the number of sequences and the number of nucleotides in each sequence. These data sets are 7(20), 9(20), 14(20), and 30(20) and contain 7, 9, 14, and 30 DNA sequences with 20 bases each. These data sets were proposed by different authors [61], [62] for solving different optimization problems. The different number of sequences enables the testing of algorithms for the best result with different kinds of instances.

\[\text{B. EVALUATION MEASURES}\]

The performance of the multi-objective and many-objective algorithms depends on the quality of the Pareto front approximation solutions [63]. The quality of solutions depends on the convergence and diversity of solutions. Multiple quality indicators are used to measure these two properties. In this paper, Hypervolume (HV) [64], Generational Distance (GD) [56], Inverse Generational Distance (IGD), and generalized spread are used. GD determines the convergence of obtained Pareto front approximation to the Pareto front. HV and IGD determine diversity and convergence. HV is the most widely used evaluation measure for the evaluation of diversity and convergence [63]. Generalized spread measures the spread of solutions in Pareto set [7]. The Euclidean distance of two neighborhood points is calculated and an average of these distances is used for the calculation of generalized spread. The goal of evolutionary algorithms is to maximize HV and minimize the values of GD, IGD, and generalized spread evaluation measures. Reference Pareto Front is composed of all non-dominated solutions, which are approximated by 10 independent runs of all the compared optimization algorithms.
C. EXPERIMENTAL SETUP

Considering the stochastic nature of multi- and many-objective algorithms, for each problem instance, 10 independent replications are run for each algorithm. The standard deviation and mean are calculated for these runs. The algorithms are implemented in JAVA and run on Intel Core i5 (1.80 GHz) with 4GB of RAM. For a fair comparison, the values of parameters of all the compared optimization algorithms are set as suggested by the authors in the corresponding papers. Table 2 shows a summary of the parametric configuration of algorithms. To perform the parametric analysis of the proposed algorithm, we have executed our algorithm on different values of polynomial mutation probability (p.m) and Simulated Binary Crossover (sbx) probability as shown in Table 3. The average values of HV, IGD, and GD metrics show that different settings of these parameters do not produce very different results.

D. RESULTS

In this section, the comparative results of different MOEAs are presented. These experiments demonstrate the effectiveness of compared algorithms for generating near-optimal solutions.

1) OPPOSITION-BASED LEARNING AND LOCAL SEARCH INFLUENCE ON RESULTS

Opposition-based Learning introduces diversity in the population, which helps to alleviate the early convergence of the algorithm. Local search is considered a famous heuristic for optimizing difficult and large problems. The Local search (LS) strategies proposed in this paper helped to obtain reliable DNA sequences. The proposed LS method improves or repairs new child solutions if required. The improvement of the newly generated child solution is considered based on Pareto dominance. The proposed Memetic GDE3 achieves better results than other algorithms used in this study because low-level heuristics like local search assist GDE3 in the exploitation of the search space more effectively. In Table 4 the results of 4 different local search strategies in GDE3 are compared. The second strategy outperforms the others, so we have used this strategy for our Memetic GDE3 when comparing with other MOEAs.

2) COMPARISON AMONG MOEAs

Comparisons are made on basis of average values of 10 independent runs. In Tables 5, 6, 7, and 8 the +, ∼, and - signs show that the values of Memetic GDE3 are significantly better (HV is greater, GD, IGD, and generalized spread are smaller), similar or worse than the current value with a confidence level of 99%, respectively. Best values are written in bold text in these tables.

In Table 5, the numerical values of average (Avg) and standard deviation (SD) of HV for 10 independent runs are presented. Considering the HV metric, Memetic GDE3 outperforms all the compared algorithms on all small (7(20), 9(20)), medium (14(20)), and large (30(20)) datasets. For example in problem 7(20), the average HV of the second-best algorithm (MOEA/D) is 0.524, while this increases to 0.542 for Memetic GDE3 (an almost 75% improvement in the quality of the obtained Pareto front). Similarly, for all other problems, we can see that there is a significant increase in the quality of the solutions obtained using Memetic GDE3. The improvement in HV reveals that the approximated Pareto front of the proposed algorithm has good convergence and diversity.

In Table 6, the numerical values of average (Avg) and standard deviation (SD) of IGD are presented for 10 independent runs. Similar to HV values, the IGD of our proposed technique of Memetic GDE3 ensures that it can generate the best results for all kinds of instances and it outperforms all the compared algorithms on all datasets. For example in problem 7(20), the average IGD of the second-best algorithms (RVEA, and MOEA/D) is 0.363, while this decreases to 0.17 for Memetic GDE3 (an almost 53% improvement in the quality of the obtained Pareto front). Similarly, for all other problems, we can see that there is a significant increase in the quality of the solutions obtained using Memetic GDE3. The improvement in IGD reveals that the approximated Pareto front of the proposed algorithm has better convergence and diversity as compared to the other algorithms.
TABLE 4. Comparison of different GDE3 local search strategies on Dataset 7(20). The results of different quality metrics are presented, which are the average of 10 runs.

| Quality Indicators | Memetic GDE3 | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|--------------------|--------------|------|----------|--------|---------|----------|------|------|
| HV                 | 0.611        | 0.137 | 0.131    | 0.137  |         |          |      |      |
| GID                | 0.009        | 0.0292| 0.0173   | 0.0164 |         |          |      |      |
| IGD                | 0.059        | 0.444 | 0.442    | 0.429  |         |          |      |      |

TABLE 5. Comparison of algorithms by using average HV. Values represent the Average (Standard Deviation). The +, ∼, and - sign show that the values of Memetic GDE3 are significantly better, similar, or worse than the current value with a confidence level of 99% respectively.

| Data Sets | MemeticGDE3 | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|-----------|--------------|------|----------|--------|---------|----------|------|------|
| 7(20)     | 0.420 (0.014)+ | 0.212 (0.024)+ | 0.133 (0.010)+ | 0.240 (0.056)+ | 0.146 (0.009)+ | 0.131 (0.015)+ | 0.131 (0.014)+ | 0.231 (0.051)+ |
| 9(20)     | 0.535 (0.013)+ | 0.085 (0.022)+ | 0.016 (0.013)+ | 0.118 (0.051)+ | 0.033 (0.028)+ | 0.017 (0.012)+ | 0.012 (0.025)+ | 0.128 (0.032)+ |
| 14(20)    | 0.333 (0.028)+ | 0.130 (0.016)+ | 0.062 (0.008)+ | 0.168 (0.027)+ | 0.079 (0.013)+ | 0.067 (0.009)+ | 0.067 (0.010)+ | 0.111 (0.048)+ |
| 30(20)    | 0.361 (0.009)+ | 0.084 (0.008)+ | 0.009 (0.004)+ | 0.107 (0.023)+ | 0.013 (0.005)+ | 0.013 (0.005)+ | 0.018 (0.007)+ | 0.024 (0.012)+ |

TABLE 6. Comparison of algorithms by using average IGD. Values represent the Average (Standard Deviation). The +, ∼, and - sign show that the values of Memetic GDE3 are significantly better, similar, or worse than the current value with a confidence level of 99% respectively.

| Data Sets | MemeticGDE3 | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|-----------|--------------|------|----------|--------|---------|----------|------|------|
| 7(20)     | 0.170 (0.004)+ | 0.388 (0.030)+ | 0.430 (0.018)+ | 0.363 (0.027)+ | 0.419 (0.026)+ | 0.427 (0.017)+ | 0.457 (0.029)+ | 0.363 (0.042)+ |
| 9(20)     | 0.119 (0.006)+ | 0.492 (0.022)+ | 0.543 (0.017)+ | 0.474 (0.032)+ | 0.520 (0.030)+ | 0.532 (0.024)+ | 0.525 (0.036)+ | 0.445 (0.033)+ |
| 14(20)    | 0.144 (0.017)+ | 0.399 (0.015)+ | 0.433 (0.017)+ | 0.370 (0.033)+ | 0.419 (0.017)+ | 0.435 (0.017)+ | 0.449 (0.017)+ | 0.437 (0.053)+ |
| 30(20)    | 0.085 (0.003)+ | 0.553 (0.016)+ | 0.586 (0.011)+ | 0.535 (0.025)+ | 0.582 (0.012)+ | 0.583 (0.025)+ | 0.582 (0.016)+ | 0.660 (0.072)+ |

TABLE 7. Comparison of algorithms by using average GD. Values represent the Average (Standard Deviation). The +, ∼, and - sign show that the values of Memetic GDE3 are significantly better, similar, or worse than the current value with a confidence level of 99% respectively.

| Data Sets | MemeticGDE3 | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|-----------|--------------|------|----------|--------|---------|----------|------|------|
| 7(20)     | 0.020 (0.001)+ | 0.023 (0.004)+ | 0.033 (0.004)+ | 0.023 (0.004)+ | 0.031 (0.002)+ | 0.032 (0.003)+ | 0.034 (0.008)+ | 0.029 (0.007)+ |
| 9(20)     | 0.014 (0.001)+ | 0.018 (0.003)+ | 0.031 (0.003)+ | 0.018 (0.005) | 0.026 (0.004)+ | 0.026 (0.002)+ | 0.027 (0.004)+ | 0.018 (0.004)+ |
| 14(20)    | 0.017 (0.003)+ | 0.012 (0.002)− | 0.021 (0.001)+ | 0.014 (0.002)− | 0.018 (0.001)− | 0.019 (0.002)− | 0.023 (0.003)− | 0.033 (0.015)+ |
| 30(20)    | 0.006 (3.7E-4)+ | 0.006 (6.2E-4)+ | 0.013 (0.002)+ | 0.009 (0.001)+ | 0.012 (8.2E-4)+ | 0.014 (0.001)+ | 0.013 (0.001)+ | 0.038 (0.006)+ |

In Table 7, the numerical values of average (Avg) and standard deviation (SD) of GD are presented for 10 independent runs. The GD of our proposed method Memetic GDE3 depicts that it can generate good results for small and large instances of the problem. However, for medium problem 14(20) which contains 14 DNA sequences with 20 bases, GDE3 and MOEA/D outperform the proposed algorithm. Moreover, the results also show that for 14(20) problem, Memetic GDE3, NSGA-II, and NSGA-III are statistically equivalent.

In Table 8, the numerical values of average (Avg) and standard deviation (SD) of generalized spread are presented for 10 independent runs. Considering generalized spread metric, Memetic GDE3 outperforms all the compared algorithms on all problem instances. For example in problem 7(20), the average generalized spread of the second-best algorithm (RVEA) is 0.732, while this decreases to 0.544 for the proposed algorithm (around 26% improvement in the quality of the obtained Pareto front). The promising results of the proposed algorithm show that the obtained set of solutions have a good spread.

Statistical comparison (p values) of algorithms with the proposed technique of Memetic GDE3 with the confidence level of 99% has been shown in Tables 9, 10, 11, and 12. Experimental results show that the proposed algorithm is statistically better than all other compared algorithms in most problem instances.
TABLE 8. Comparison of algorithms by using average Generalized Spread. Values represent the Average (Standard Deviation). The +, ~, and - sign show that the values of Memetic GDE3 are significantly better, similar, or worse than the current value with a confidence level of 99% respectively.

| Data Sets | MemeticGDE3 | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|-----------|-------------|------|----------|--------|---------|----------|------|------|
| 7(20)     | 0.544 (0.045) | 0.811 (0.086) + | 0.790 (0.083) + | 0.759 (0.075) + | 0.767 (0.072) + | 0.763 (0.041) + | 0.820 (0.067) + | 0.752 (0.066) + |
| 9(20)     | 0.510 (0.043) | 0.800 (0.066) + | 0.754 (0.055) + | 0.789 (0.034) + | 0.781 (0.034) + | 0.770 (0.049) + | 0.799 (0.088) + | 0.775 (0.058) + |
| 14(20)    | 0.579 (0.048) | 0.804 (0.046) + | 0.721 (0.053) + | 0.713 (0.050) + | 0.723 (0.064) + | 0.718 (0.049) + | 0.787 (0.050) + | 0.804 (0.051) + |
| 30(20)    | 0.522 (0.044) | 0.760 (0.031) + | 0.705 (0.052) + | 0.763 (0.034) + | 0.670 (0.032) + | 0.691 (0.031) + | 0.749 (0.025) + | 0.876 (0.035) + |

TABLE 9. 1-tailed comparison of algorithms HV values by using p-value of T-test with Memetic GDE3.

| Data sets | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|-----------|------|----------|--------|---------|----------|------|------|
| 7(20)     | <0.001 | 0 | <0.001 | 0 | 0 | 0 |
| 9(20)     | <0.001 | 0 | <0.001 | <0.001 | 0 | 0 | 0 |
| 14(20)    | <0.001 | 0 | <0.001 | <0.001 | 0 | <0.001 | 0 |
| 30(20)    | 0 | 0 | <0.001 | 0 | 0 | 0 | 0 |

TABLE 10. 1-tailed comparison among algorithms IGD values by using p-value of T-test with Memetic GDE3.

| Data sets | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|-----------|------|----------|--------|---------|----------|------|------|
| 7(20)     | 0.004 | <0.001 | <0.001 | 0.008 | <0.001 | 0 | <0.001 |
| 9(20)     | <0.001 | <0.001 | 0.03 | <0.001 | <0.001 | 0 | <0.001 |
| 14(20)    | <0.001 | <0.001 | 0.03 | <0.001 | <0.001 | 0 | <0.001 |
| 30(20)    | 0.5 | 0.497 | 0.499 | 0.499 | 0.497 | 0.498 | 0.489 |

TABLE 11. 1-tailed comparison among algorithms GD values by using p-value of T-test with Memetic GDE3.

| Data sets | GDE3 | MaOEA/AC | MOEA/D | NSGA-II | NSGA-III | DBEA | RVEA |
|-----------|------|----------|--------|---------|----------|------|------|
| 7(20)     | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 |
| 9(20)     | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 |
| 14(20)    | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 |
| 30(20)    | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 | <0.001 |

TABLE 12. 1-tailed comparison among algorithms generalized spread values by using p-value of T-test with Memetic GDE3.

of the cases. The results of HV and IGD metrics (both are used to measure both convergence and diversity simultaneously) reveal that the proposed algorithm has better performance than all the compared meta-heuristics in all instances. Moreover, the proposed many-objective memetic GDE3 outperforms all other compared meta-heuristics in all problem instances and obtains well-distributed solutions on the whole front. However, in the case of GD, which measures only the convergence property of the algorithm, RVEA and MOEA/D give better results on a medium-sized problem as compared to Memetic GDE3.

Figure 5 shows box plots of HV and IGD measures of compared algorithms for the smallest (7(20)) and the largest problem (30(20)). Box plots show a graphical representation of spread and skewness in numerical data. The horizontal lines inside the boxes show the medians of the values. The size of the box plot indicates variance in different runs of the algorithm. As it can be seen from Figure 5, Memetic GDE3 consistently shows very competitive results for all datasets. The plots also show that Memetic GDE3 has good stability as it has close-packed boxes compared to most of the other algorithms.

Besides the discussed qualities, Memetic GDE3 has some limitations, which need to be mentioned. First, the conventional differential evolution-based meta-heuristics have some limitations in converging to the global optimal due to the difficult and time-consuming task of tuning different parameters [65]. Second, the results of the GD metric (used to measure convergence) also show that the convergence property of the proposed meta-heuristic is compromised for some
problem instances due to enhanced exploratory behavior of the opposition-based learning.

VII. CONCLUSION AND FUTURE WORK

In this study, designing sequences of DNA is formulated as a MaOP, and an opposition-based memetic GDE3 algorithm is proposed to solve the problem. The effectiveness of the proposed approach is assessed using 4 datasets of different sizes, and 7 popular algorithms. Evaluation is done by using 4 frequently used quality indicators, which are HV, GD, IGD, and Generalized Spread. Statistical significance tests show that our proposed Memetic GDE3 outperforms other well-known multi-objective algorithms like NSGA-III, NSGA-II, MOEA/D, GDE3, RVEA, DBEA, and a recently proposed many-objective algorithm MaOEA/AC. One of the possible future directions is to use hyper-heuristics, which solve the optimization problems using a combination of different meta-heuristics. Chaotic strategies can also be incorporated into many-objective evolutionary algorithms to improve the design [66]. Many-objective
References
