Dynamic Group-Based Cooperative Optimization Algorithm

MOHAMAD M. FOUAD1, (Student Member, IEEE), ALI IBRAHIM EL-DESOUKY1, RAMI AL-HAJJ2, (Member, IEEE), AND EL-SAYED M. EL-KENAWY3, (Member, IEEE)

1Department of Computer Engineering and Systems, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt
2College of Engineering and Technology, American University of the Middle East, Egaila, Kuwait
3Department of Communications and Electronics, Delta Higher Institute for Engineering and Technology, Mansoura 35111, Egypt

Corresponding author: Mohamad M. Fouad (mfouad@ieee.org)

ABSTRACT Several optimization problems from various types of applications have been efficiently resolved using available meta-heuristic algorithms such as Particle Swarm Optimization and Genetic Algorithm. Recently, many meta-heuristic optimization techniques have been extensively reported in the literature. Nevertheless, there is still room for new optimization techniques and strategies since, according to the literature, there is no meta-heuristic optimization algorithm that may be considered as the best choice to cope with all modern optimization problems. This paper introduces a novel meta-heuristic optimization algorithm named Dynamic Group-based Optimization Algorithm (DGCO). The proposed algorithm is inspired by the cooperative behavior adopted by swarm individuals to achieve their global goals. DGCO has been validated and tested against twenty-three mathematical optimization problems, and the results have been verified by a comparative study with respect to state-of-the-art optimization algorithms that are already available. The results have shown the high exploration capabilities of DGCO as well as its ability to avoid local optima. Moreover, the performance of DGCO has also been verified against five constrained engineering design problems. The results demonstrate the competitive performance and capabilities of DGCO with respect to well-known state-of-the-art meta-heuristic optimization algorithms. Finally, a sensitivity analysis is performed to study the effect of different parameters on the performance of the DGCO algorithm.

INDEX TERMS Optimization, meta-heuristics, exploration, exploitation, evolutionary optimization, DGCO.

I. INTRODUCTION

Nowadays, real-world optimization problems are complex with high dimensional search space therefore, they are in general challenging. Heuristic Optimization techniques have been applied in several fields such as engineering [1], machine learning [2], business processes [3], mechanics [4], economics [5], scheduling [6], transportation [7], integrated decision making [8], and formulas estimation [9], [10].

Optimization refers to finding an acceptable optimal solution for a specific problem among many feasible ones. An optimization task is usually transformed into a search problem in a multi-dimensional space. Practically, that search refers to minimizing or maximizing an objective function that evaluates the quality of a solution candidate that is usually denoted by a vector in the search space. Meta-heuristics are a family of approximate optimization techniques that provide acceptable solutions in a reasonable time [11]. They are adopted for solving hard and complex problems in science and engineering.

Meta-heuristic optimization algorithms have become more popular because of their simplicity and flexibility with respect to classical and exact optimization methods like Greedy Search and Local Search [12]. That simplicity is due to the fact that they are based on simple concepts that are easy to understand and implement. In general, those meta-heuristic optimization algorithms are flexible because they can be applied in various fields and applications without special changes to their design and implementation. Moreover, they are able to avoid local optima because of their stochastic nature that allows them to extensively explore the search space and avoid stagnation in local optima. In addition, meta-heuristics are derivative-free techniques, so they do not need to use derivative information of the search space compared to gradient-based algorithms. In fact, a search space in real-world problems is often very complex with an enormous number of local optima and expensive or unknown derivative
information, so meta-heuristics are more suitable for optimizing those types of problems.

Local search optimization algorithms such as Tabu Search [13] and Simulated Annealing (SA) [14] use a single randomly initialized solution that is improved over the course of iterations. On the other hand, population-based meta-heuristics start the optimization process with an initial random group of individuals (population) that represent candidate solutions to the problem. The individuals exchange information about the search space and cooperate to avoid local optima and converge towards a global goal.

In fact, the quality and effectiveness of a meta-heuristic algorithm depend on its ability to ensure an appropriate balance between exploration of the search space (diversification) and exploitation of the best solutions found (intensification) [15]. Exploration is achieved by the meta-heuristic in order to discover new areas that may contain promising points in the search space. On the other hand, exploitation is the process of finding better solutions around the good solutions found so far. Most of the meta-heuristics perform more explorations at the initial stages to extensively explore the search space and to avoid stagnation in local optima. The main difference among meta-heuristic algorithms is the technique they ensure a balance between exploration and exploitation.

Evolutionary algorithms are one type of population-based meta-heuristics that are inspired by the process and mechanism of biological evolution. The most popular evolutionary algorithm is the Genetic Algorithm (GA) [16] that was proposed by John Holland and is inspired by the process of natural selection. It uses genetic operators such as selection, crossover, and mutation to evolve an initial random population throughout generations. GA starts with an initial random population. Then, for each generation, the fitness is calculated for all individuals in the population, and parents are selected for mating and creating offspring. The mutation is applied to guarantee the diversity of the population. Other popular evolutionary algorithms are Evolutionary Programming (EP) [17], Evolution Strategy (ES) [18], and Differential evolution (DE) [19].

Swarm-based algorithms are another type of meta-heuristic algorithms that are inspired by the social behavior of swarm systems in nature. Particle Swarm Optimization (PSO) [20] is the most popular swarm-based algorithm. PSO mimics the movement and interaction among individuals in a bird flock or fish school. Each candidate solution is called a particle and its movement is influenced by its own experience and the best solution found by the swarm. Ant Colony Optimization (ACO) [21] is another popular swarm-based algorithm that is inspired by the foraging behavior of some ant species. Grey Wolf Optimizer (GWO) [22] is also another interesting example of the swarm-based methods that mimic the social hierarchy and hunting behavior of grey wolves. Other examples of swarm-based algorithms are Artificial Bee Colony (ABC) [23], Firefly Algorithm [24], Cuckoo Search [25], Whale Optimization Algorithm (WOA) [26], and Bat Algorithm (BA) [27].

Unlike evolutionary algorithms and swarm-based algorithms, some meta-heuristics are inspired by laws of physics [28]. These include Gravitational Search Algorithm (GSA) [29] that is inspired by Newtonian gravity rule, Simulated Annealing (SA) [14], and Intelligent Water Drops Algorithm (IWD) [30].

Table 1 lists some popular meta-heuristics along with their working principles, advantages, limitations, and potential applications. These algorithms have been selected to be compared with our proposed algorithm because of their popularity and their similar processing principles. As illustrated in Table 1, some algorithms are inspired by the laws of biological evolution while others are inspired by the foraging or the hunting behaviors of some animals. All of these algorithms are gradient-free which means that they do not require any gradient information of the search space. Most of these algorithms have been extensively used to solve several optimization problems because of their advantages such as ease of understanding and implementation, fast convergence, and large-scale exploration. However, most of them tend to converge prematurely due to stagnation in local optima. Other algorithms have slow convergence speed or low diversity among their candidate solutions. As a result, these algorithms have different variations that improve some features of the original algorithms such as convergence speed, exploration behavior, and the trade-off between exploration and exploitation [12], [31], [32].

II. LITERATURE REVIEW

In real applications, the search space is generally complex and subject to many local optima. This increases the likelihood of stagnation into local optima therefore causing premature convergence of an optimization algorithm. Most optimization algorithms try to solve this challenge by including techniques that increase the diversity of the population [33]. These techniques may help in avoiding local optima; however, they may affect the convergence performance. Hence, a good balance between exploration and exploitation is required to develop a performant meta-heuristic algorithm for optimization [34]. This balance improves the convergence speed of the optimization algorithm and provides a higher exploration of the search space with the ability to avoid local optima.

Recently, many meta-heuristic algorithms have been introduced in the literature with different techniques to tackle these common challenges. For instance, Butterfly Optimization Algorithm (BOA) [35] is a nature-inspired meta-heuristic that has been recently introduced. BOA mimics the food search and mating behavior of butterflies. BOA uses simple strategies to achieve exploration and exploitation. In BOA, the butterfly may move randomly in the search space to achieve exploration or move towards the best butterfly to achieve exploitation. The balance between exploration and
TABLE 1. List of some popular meta-heuristics developer in the literature.

| Optimizer | working principles | Advantages | Limitations | Applications |
|-----------|---------------------|------------|-------------|--------------|
| Genetic Algorithm (GA) [16], [44], [45] | GA is inspired by the theory of biological evolution (survival of the fittest). The main steps of GA are initialization, parent selection, crossover, and mutation. Other genetic operators may be applied such as elitism. | Easy to understand and implement. Gradient-free method. It can handle noisy fitness functions. | Slow convergence time. Premature convergence. GA parameter tuning. It does not scale well with complexity. | Antennas design. Scheduling. Robot path planning. Engineering applications. Signal processing — filter design. |
| Particle Swarm Optimization (PSO) [20], [32], [46] | PSO mimics movements and interactions among individuals in a bird flock or fish school. The movement of each individual is inspired by its own known best position and best position obtained by the swarm. | Stable convergence. Easy to implement. Good cooperation model. | Difficult to determine initial parameters. High computational cost. Slow convergence time. | Gene clustering. Antenna design. Vehicle routing. Dimensionality reduction. Design of controllers. |
| Differential Evolution (DE) [19], [47], [48] | DE is inspired by the principle of survival of the fittest. DE starts with an initial random population. Then it creates new candidates by combining existing ones with mutant solutions. Finally, DE keeps solutions with better scores. | Easy to understand. Robust convergence. Encourage diversity of the population. Robust to changing circumstances. | Slow convergence time. High computational cost. Difficult to determine the parameters of the algorithm. | Solve the optical power flow. Capacitive voltage divider. Antenna array design. Predict protein folding. Feature extraction. |
| Grey Wolf Optimizer (GWO) [22], [49], [50] | GWO mimics the leadership and social hierarchy and hunting behavior of the grey wolves. Individuals of the pack follow three leaders towards their prey. The main steps are encircling prey, hunting, and attacking the prey. | Good balance between exploration and exploitation. High search precision. Simple and easy to implement. | Premature convergence because the three leaders update their position based on each other. The performance degrades proportionally to the number of variables. | Feature selection. Adjustment of PID controller parameters in DC motor. Clustering applications Robotics and pathfinding. |
| Whale Optimization Algorithm (WOA) [37], [51], [52] | WOA is inspired by the bubble-net foraging behavior of humpback whales. They use bubbles to trap fishes as they move in a spiral-shaped path around a school of fish. | Large-scale exploration of the search space. Easy to implement. | Slow convergence speed. Stagnation into local optima. High computational cost. | Offset voltage minimization. Route optimization. Task Scheduling Accuracy control of laser sensor system. |

exploitation depends on a switch probability. BOA has been validated on classical benchmark functions and engineering design problems. In general, BOA exhibits good performance and results.
Another example of recently introduced meta-heuristics is the Stochastic Fractal Search (SFS) that is inspired by the natural phenomenon of growth [36]. SFS applies two main processes during the optimization task: the diffusing process and the updating process. The first process ensures exploitation while the updating process increases the exploration of the search space. Moreover, SFS uses two strategies to create new particles: Levy flight and Gaussian. These strategies help increase the convergence speed of the algorithm. SFS has been tested against both constrained and unconstrained standard benchmark functions, and it showed good performance and high exploration capabilities.

Whale Optimization Algorithm WOA [37] applies different strategies to achieve exploration and exploitation. Some solutions move around a randomly selected solution to improve exploration. On the contrary, other solutions move towards the best solution in a spiral movement to satisfy exploitation. WOA depends on two adaptive parameters to achieve the balance between exploration and exploitation. WOA has been tested and validated against standard benchmark functions and constrained engineering design problems. Moreover, many variants of WOA have been introduced and applied to several optimization problems.

Another newly introduced optimizer is the Harris Hawks Optimization (HHO) [38] which is inspired by the hunting behavior of Harris Hawks. The exploration phase mimics the way hawks search for prey by perching in random locations and waiting to detect prey. In the exploitation phase, HHO uses four strategies to attack the prey. HHO moves from exploitation to exploration through iterations using an adaptive equation that is similar to the one used in WOA. HHO has been tested and validated against several benchmark functions and constrained engineering design problems. HHO proved to be promising and competitive.

Recently, many researchers have introduced hybrid optimization algorithms that combine the advantages of two or more optimization algorithms to overcome the limitations of a single optimization algorithm. For instance, in [39] the PSO algorithm is combined with the Sine Cosine Algorithm and levy flight approach resulting in a new hybrid optimizer called PSOSCA. The levy flight approach introduces random walks in the search space. These random walks guarantee high exploration and better avoidance of local optima. The position updating equations in the Sine Cosine Algorithm (SCA) [40] enhance the exploration and exploitation capabilities of PSO. The new hybrid PSOSCA has been validated using standard benchmark functions and real constrained engineering problems. PSOSCA showed its advantages and effectiveness against most of the PSO variants.

In [41], a new variant of the Bat Algorithm (BA) [27] called ASF-BA was introduced. ASF-BA added adaptive inertia weight to speed up the velocity rate of bats. This technique improves the diversity among the individuals in the population. Moreover, ASF-BA replaces the weak random searching method of the Bat algorithm with the Sugeno function for fuzzy search that enhances the local search ability of bats.

The new hybrid algorithm ASF-BA has been validated and tested on several benchmark functions and proved to be promising in terms of convergence speed, stability, and solution quality.

In [42], the authors tested the hybridized Genetic Algorithm and Simulated Annealing (IGASA) to get rid of the limitations of GA and SA. The hybrid algorithm (IGASA) has been used for solving both low-dimensional and high-dimensional knapsack problems. The reported results proved that IGASA offered good quality solutions. However, it is more computationally expensive than GA and SA.

According to the No Free Lunch (NFL) theorem [43], there is no meta-heuristic best suited for solving all optimization problems. This explains why certain meta-heuristics perform better on specific optimization problems and not as good on others. Consequently, new optimizations are still being proposed. Our proposed optimization algorithm presents techniques to handle some of the drawbacks of existing optimization algorithms such as slow convergence, the balance between exploration and exploitation, and stagnation into local optima. The NFL theorem and attempts to overcome existing drawbacks have motivated us to develop the proposed optimization algorithm.

In this paper, we propose a new meta-heuristic optimization method, named Dynamic Group-based Cooperative Optimization Algorithm (DGCO) that is inspired by the cooperative behavior of swarm individuals to achieve their global goals. DGCO attempts to strike a balance between ensuring fast convergence and avoiding stagnation between local optima. This is implemented by applying techniques that help in enhancing the performance of exploitation, achieving a good balance between exploration and exploitation, enhancing the exploration of the search space, and increasing the diversity among the individuals of a current population. The main contribution of this paper is the introduction of DGCO as a new optimization algorithm with new insight into solving different optimization problems. Preliminary studies show that DGCO is competitive, promising, and capable of outperforming existing swarm-based and evolutionary-based algorithms. Moreover, the performance of the proposed algorithm has been verified on constrained engineering design problems.

The rest of this paper is organized as follows. An introduction to DGCO is given in Section III. The experimental results of validating and testing our approach against mathematical benchmark functions and constrained engineering design problems are described in Section IV and Section V respectively. A sensitivity analysis of the parameters of DGCO and their effect on the performance is introduced in Section VI. Finally, in Section VII, some conclusions and potential future work are presented.

III. DGCO OPTIMIZATION ALGORITHM

A. MOTIVATIONS

DGCO is a dynamic group-based cooperative optimization algorithm that mimics the cooperation among individuals in
a swarm to achieve a global goal of optimization. In nature, creatures tend to live in groups and communities. They usually collect food and defend against enemies together in a cooperative way while exchanging roles when achieving their tasks. They arrange themselves into sub-groups among which the individuals tend to inter-cooperate in the same sub-group and with individuals of other groups to achieve their global goal. For instance, ant colonies and bee colonies are the most popular examples of cooperation among individuals in a swarm. Each member in the swarm has a specific role to do for the colony. Soldier members are responsible for defending the colony whereas worker members are responsible for finding food and feeding other members. DGCO is inspired by the fact that members in groups tend to be arranged into sub-groups to interchangeably perform different duties at a period and cooperate to achieve their common goals. The solution of an optimization problem consists of resolving two sub-tasks: Exploration and Exploitation. In our case, DGCO divides individuals, also named search agents, into two sub-groups, where each is dedicated to solving one of those two complementary sub-tasks. The number of individuals in each sub-group is dynamically controlled by DGCO. Each group applies two different techniques to complete its specific task, as described in the following sections. Furthermore, DGCO’s techniques of exploration and exploitation ensure a good exploration of the search space, encourage diversity, preserve convergence, and avoid stagnation in local optima. In most of the cooperative optimization algorithms, all individuals perform exploitation at final iterations that may cause stagnation into local optima. DGCO avoids this phenomenon by keeping a group of search agents doing exploration over the course of iterations. Moreover, DGCO increases the number of exploring individuals immediately if the performance of the algorithm does not show enhancement in any three consecutive iterations.

The individuals that constitute a population represent solutions candidates of an optimization problem. Each of these individuals consists of a vector of parameters. The population of candidate individuals is divided into two sub-groups: the exploration group and the exploitation group. The individuals in the exploration group mainly focus on exploring new areas of the search space for an optimal solution, whereas the individuals of the second group mainly focus on improving the quality of the actual best solution found based on an objective function. Unlike the traditional evolutionary algorithms, the individuals of DGCO that belong to the two sub-populations cooperate among themselves and exchange information and duties to ensure two goals: (1) good exploration of the search space, and (2) avoid local optima in an efficient way. DGCO keeps control of the balance between exploration and exploitation and offers an automatic mechanism to avoid the steady regions of the search space.

**B. BASIC CONCEPTS AND FORMULATION**

An optimization problem in the science and engineering domain consists of finding the best solution to a problem within a set of given constraints and conditions. Such a problem can be considered as a search problem for an optimal solution that can be represented by a vector in a search space. In DGCO, an individual of the population, belonging to either the exploration group or the exploitation group, can be represented by a vector \( \mathbf{S} = \{S_1, S_2, \ldots, S_d\} \in \mathbb{R}_d \) where \( S_i \) represents a parameter of the system or the problem to be solved or optimized, and \( d \) is the dimension of the problem and therefore of the search space. To assess how much an individual is performant till a certain moment, a fitness function \( f \) is adopted in the proposed algorithm for that purpose. The optimization process evolves through populations in looking for a certain optimal vector \( S^* \) that optimizes the fitness function using the following steps:

1) **INITIALIZATION**

The algorithm starts with initial random individuals (solutions). The initial value of each component \( p_i \) in a \( d \)-dimensional vector belongs to a range \([\min_p, \max_p]\) specific to the parameter represented in that vector. Therefore, DGCO requires the following parameters to start the optimization process: (1) the number of solutions that will be denoted by the population size, (2) the dimension of each solution, (3) the lower bound and upper bound of each solution component, and (4) the fitness function.

2) **DYNAMIC GROUP-BASED COOPERATION**

After initialization, a fitness value is calculated for each solution in the population. Then, the algorithm finds the best solution that has an optimal fitness value. After that, the algorithm divides the population’s individuals into two groups: the exploration group and the exploitation group. DGCO ensures the balance between exploration and exploitation by dynamically changing the number of individuals in each group gradually. Each group applies two different strategies to achieve their global goal. Figure 1 illustrates the two groups of the DGCO algorithm.

For instance, some individuals in the exploitation group move towards the best solution designated by “leader”, and other individuals search in the area around the leader. For the exploration group, an individual, up to a certain probability, performs a mutation in one or more of its parameter values, and other individuals search in the area around themselves to search for promising areas in the search space. Then, DGCO randomly exchanges the individuals between the two groups to ensure a certain degree of randomness. These techniques will be discussed in detail in the following sections.

**C. BALANCING BETWEEN EXPLORATION AND EXPLOITATION**

To ensure a balance between exploration and exploitation, DGCO dynamically changes the number of individuals in each of the population’s sub-groups. Initially, the algorithm starts with a (70/30) schema by assigning 70% of the population’s individuals to the exploration group whereas the remaining 30% are assigned to the exploitation group. It is
worth mentioning that starting with a high percentage of individuals in the exploration group helps in exploring more promising areas in the search space in the early stages of the optimization process. The number of individuals in the exploration group decreases dynamically from 70% to 30%, whereas the number of individuals in the exploitation group increases from 30% to 70% over the course of iterations allowing more improvement in the global average of individuals’ fitness by letting more exploiting individuals improve their fitness value. Moreover, the algorithm applies the elitism technique in order to ensure the convergence of the process by keeping the leader individual in a subsequent population in case no better solution is found for that new population. In order to avoid local optima and stagnation problems, DGCO may increase the number of individuals in the exploration group at any iteration if the fitness of the leader does not improve significantly for three consecutive iterations.

Figure 2 shows how the number of individuals in exploration and exploitation groups change dynamically over the course of iterations. Figure 2(a) shows a sample convergence curve for a sample optimization problem of finding a point in 2-dimensional space. Figure 2(b) illustrates the same process described in (a) depicting the complementary number of exploration and exploitation of individuals over the iterations of the process.

**D. EXPLORATION GROUP**

The exploration group is responsible not only for finding promising areas in the search space but also for avoiding stagnation in local optima. To achieve that, DGCO uses two different techniques for exploration: explore around the current individual and mutation.

1) **EXPLORE SEARCH SPACE AROUND THE SOLUTION**

In this technique, the individual is looking for promising areas around its location in the search space. This is performed by iteratively searching for a better solution in terms of fitness value among its neighboring possible solutions. For this purpose, DGCO uses the following equations:

\[
\vec{D} = r_1 \cdot (\vec{S}(t) - 1) \\
\vec{S}(t + 1) = \vec{S}(t) + \vec{D}.(2r_2 - 1)
\]

where \(r_1\) and \(r_2\) are coefficient vectors in the intervals [0, 2] and [0, 1] respectively, \(t\) refers to the current iteration, \(\vec{S}\) is the current solution vector, and \(\vec{D}\) indicates the diameter of the circle in which the solution will look for promising areas.

2) **MUTATION OF THE SOLUTION**

Another technique that DGCO applies for exploration is mutation. It is a genetic operator used to introduce and maintain diversity in the population. It can be seen as a
local random perturbation of one or more components in the individuals based on a certain probability. It helps avoid local optima thus preventing premature convergence. Such a perturbation acts as a jump to another promising area in the search space. In fact, mutation is one of the key factors that gives DGCO its high exploration capability.

E. EXPLOITATION GROUP
The exploitation group is responsible for obtaining even better solutions from good ones. At the beginning of each iteration, DGCO calculates the fitness value of all individuals and recognizes the best individual that has the best fitness value. In order to achieve exploitation, DGCO uses two different techniques, as follows:

1) MOVING TOWARDS THE BEST SOLUTION
In this technique, the individual moves towards the best solution using the following formulas:

\[ \vec{D} = \vec{r_3} \cdot (\vec{L}(t) - \vec{S}(t)) \]  
\[ \vec{S}(t + 1) = \vec{S}(t) + \vec{D} \]  

where \( \vec{r_3} \) is a random vector in the interval \([0, 2]\) that controls the step of the movement towards the leader solution, \( t \) refers to the current iteration, \( \vec{S} \) is the vector of the current solution, \( \vec{L} \) is the vector of the best solution, and \( \vec{D} \) indicates the distance vector.

2) SEARCHING AROUND THE BEST SOLUTION
The area around the best solution (leader) is most likely to be promising. Therefore, some individuals search in the area around the best solution with the hope to find an even better solution. To achieve this, DGCO uses the following formulas:

\[ \vec{D} = \vec{L}(t) \ast (\vec{k} - \vec{r_4}) \]  
\[ \vec{S}(t + 1) = \vec{S}(t) + \vec{D} \cdot (2\vec{r_5} - 1) \]  
\[ \vec{k} = 2 - \frac{2 \times t^2}{\text{iters}_\text{count}^2} \]  

where \( \vec{r_4} \) and \( \vec{r_5} \) are random vectors in the interval \([0, 1]\), \( \vec{k} \) decreases exponentially from 2 to 0 over the course of iterations, \( \vec{L} \) is the vector of the best solution, \( \vec{S} \) is the current solution vector, and \( \vec{D} \) indicates the diameter of the circle in which the solution will look for better solutions.

F. ELITISM OF THE BEST SOLUTION
To guarantee the solution’s quality, DGCO elects the best solution to the next iteration of the process without any modifications. The elitism strategy improves the performance of the algorithm; however, it may lead to premature convergence in multimodal functions [53], [54]. It is worth mentioning that DGCO has high exploration capabilities by using mutation strategy and searching around individuals in the exploration group. This high exploration capability helps DGCO prevent premature convergence.
The pseudo-code of the DGCO algorithm is represented in Algorithm 1. First, we provide DGCO with input parameters: population size, iterations count, and mutation rate. Then, DGCO divides the individuals into two groups: the exploration group and the exploitation group. The algorithm controls the number of individuals in each group dynamically. Each group applies two different strategies to achieve their tasks. It is worth mentioning that DGCO randomizes the order of solutions at the end of each iteration to guarantee diversity and high exploration. For instance, a solution in the exploration group in an iteration may be a member of the exploitation group in the next iteration. The elitism strategy applied in DGCO prevents the algorithm from losing the leader individual from one iteration to the next.

The main execution steps of the proposed algorithm are the following:

- **Initialization:** The algorithm randomly generates a predefined number of individuals. Each individual represents a candidate solution to the problem being solved.
- **Fitness Evaluation:** Each candidate solution is evaluated using a fitness function that measures the quality of the solution.
- **DGCO Groups:** The algorithm divides the individuals of the population into two groups and then manages the number of solutions of each group dynamically at the beginning of each iteration. Over the course of iterations, the number of individuals in the exploration group decreases from 70% to 30% of the total number of individuals. On the other hand, if the fitness values of the leaders in three consequent iterations do not change significantly, the algorithm immediately increases the number of individuals in the exploration group to become 70% of the total number of individuals with the hope of finding other promising areas in the search space or avoiding stagnation in local optima.
- **Exploration / Exploitation:** DGCO applies two different strategies to explore the search space in an efficient way. These strategies are search around the solution and mutation. Also, it applies two different strategies to achieve exploitation. These strategies are moving toward the best solution and searching in the area around the leader solution.
- In the end, DGCO modifies the individuals that go beyond the search space. Then, the order of individuals is changed randomly in order to exchange the roles of members of the exploration and exploitation groups. At the end of the optimization, DGCO returns the best solution.

**Algorithm 1:** Pseudocode of DGCO Optimization Algorithm

```plaintext
1: Input parameters (population size, iterations count, mutation rate)
2: Initialize the population \( \vec{S} = \{S_1, S_2, \ldots, S_d\} \)
3: 
4: While \( t < \text{iters\_count} \)
5: Calculate fitness of each solution, calculate best solution
6: \( k = 2 - 2 \times t / \text{iter\_count} \)
7: 
8: Update the number of solutions in each group
9: if best fitness didn’t change from previous 2 iterations
10: Increase the number of solutions in the exploration group
11: end if
12: 
13: for each solution in the exploration group
14: update \( r_1, r_2, \) and \( p \)
15: elitism of the best solution
16: if \( p >= 0.5 \)
17: mutate the solution
18: else
19: search around current solution (eq. 2)
20: end if
21: end for
22: 
23: for each solution in the exploitation group
24: elitism of the best solution
25: update \( r_2, r_3, r_4, \) and \( p \)
26: if \( p >= 0.5 \)
27: move towards the best solution (eq. 4)
28: else
29: search around the best solution (eq. 6)
30: end if
31: end for
32: 
33: amend solutions that go beyond the search space
34: update prev\_fitness1, prev\_fitness2
35: end while
36: return Best solution
```

**IV. RESULTS AND DISCUSSION**

In order to evaluate the performance of the proposed optimization algorithm, twenty-three standard benchmark mathematical functions have been used to find their minimum values in a specific domain of the search space. These functions are widely used in literature for benchmarking optimization algorithms [55]. Those functions are divided into three categories: Unimodal functions, Multimodal functions, and Multimodal functions with fixed dimensions. Table 2 and Table 3 list the benchmark functions where \( D \) represents the dimension of the function and \( f_{\text{min}} \) represents the optimum value of the function that is the minimum in this case.

Unimodal high-dimensional functions are used to test the convergence rate of the search algorithm. These functions are suitable for benchmarking exploitation performance. On the other hand, Multimodal high-dimensional functions are more difficult to optimize as they have many local optima, so they are suitable to benchmark the exploration capabilities of
TABLE 2. Descriptions of unimodal benchmark functions used in our experiments.

| Benchmark Function | D  | Range     | $f_{\min}$  |
|--------------------|----|-----------|--------------|
| $f_{01}(x) = \sum_{i=1}^{n} x_i^2$ | 30 | [-100, 100] | 0            |
| $f_{02}(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$ | 30 | [-10, 10] | 0            |
| $f_{03}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$ | 30 | [-100, 100] | 0            |
| $f_{04}(x) = \max_i \{ |x_i|, 1 \leq i \leq D \}$ | 30 | [-100, 100] | 0            |
| $f_{05}(x) = \sum_{i=1}^{D} [100(x_i+1 - x_i^2)^2 + (x_i-1)^2]$ | 30 | [-30, 30] | 0            |
| $f_{06}(x) = \sum_{i=1}^{D} (|x_i| + 0.5)^2$ | 30 | [-100, 100] | 0            |
| $f_{07}(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1]$ | 30 | [-1.28, 1.28] | 0            |

TABLE 3. Descriptions of multimodal benchmark functions used in our experiments.

| Benchmark Function | D  | Range            | $f_{\min}$  |
|--------------------|----|------------------|--------------|
| $f_{08}(x) = \sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|})$ | 30 | [-500, 500] | -12569.487 |
| $f_{09}(x) = \sum_{i=1}^{D} [x_i^2 - 10 \cos(2\pi x_i) + 10]$ | 30 | [-5.12, 5.12] | 0            |
| $f_{10}(x) = -20 \exp \left( -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^{D} \cos 2\pi x_i \right) + 20 + e$ | 30 | [-32, 32] | 0            |
| $f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos \left( \frac{x_i}{\sqrt{10}} \right) + 1$ | 30 | [-600, 600] | 0            |
| $f_{12}(x) = \frac{\pi}{D} \left[ 10 \sin^2(\pi y_1) + \sum_{i=1}^{D} (y_i - 1)^2 \left[ 1 + 10 \sin^2(\pi y_i + 1) + (y_D - 1)^2 + \sum_{i=1}^{D} u(x_i, 4, 10, 100, 4) \right] \right]$ | 30 | [-50, 50] | 0            |
| $f_{13}(x) = 0.1 \left[ 10 \sin^2(3\pi y_1) + \sum_{i=1}^{D} (y_i - 1)^2 \left[ 1 + 10 \sin^2(3\pi y_i + 1) \right] + (x_n - 1)^2 \left[ 1 + \sin^2(2\pi x_n) \right] + \sum_{i=1}^{D} u(x_i, 5, 100, 4) \right]$ | 30 | [-50, 50] | 0            |
| $f_{14}(x) = \left( \frac{1}{4000} + \sum_{i=1}^{D} \left( \frac{1}{4000} f_i + \frac{1}{4000} \sum_{i=1}^{D} (x_i - a_i)^2 \right) \right)^{-1}$ | 2 | [-65, 65] | 1            |
| $f_{15}(x) = \sum_{i=1}^{D} \left( x_i - \frac{x_i^4 + b_i x_i + c_i}{4} \right)^2$ | 4 | [-5, 5] | 0.00030     |
| $f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3} x_1^6 + x_1^2 x_2 - 4x_2^2 + 4x_2^4$ | 2 | [-5, 5] | -1.0316     |
| $f_{17}(x) = \left( x_2 - \frac{5}{4000} x_1^2 + \frac{5}{4000} x_1 - 6 \right)^2 + 10 \left( 1 - \frac{1}{8\pi} \right) \cos x_1 + 10$ | 2 | [-5, 5] | 0.398       |
| $f_{18}(x) = \left[ 1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 2x_1 x_2 + 3x_2^2) \right] \times \left[ 30 + 2(x_1 - 3x_2 x_1^2 x_2 + 12x_2^2 + 48x_2 - 36x_2 x_2 + 27x_2^2) \right]$ | 2 | [-2, 2] | 3           |
| $f_{19}(x) = -\sum_{i=1}^{D} c_i \exp \left( -\frac{1}{4} \sum_{i=1}^{D} a_{ij} x_i - p_{ij} \right)^2$ | 3 | [1, 3] | -3.86       |
| $f_{20}(x) = -\sum_{i=1}^{D} c_i \exp \left( -\sum_{j=1}^{D} a_{ij} x_i - p_{ij} \right)^2$ | 6 | [0, 1] | -3.32       |
| $f_{21}(x) = -\sum_{i=1}^{D} (x - a_i)(x - a_i)^T + c_i^{-1}$ | 4 | [0, 10] | -10.1532    |
| $f_{22}(x) = -\sum_{i=1}^{D} (x - a_i)(x - a_i)^T + c_i^{-1}$ | 4 | [0, 10] | -10.4028    |
| $f_{23}(x) = -\sum_{i=1}^{D} (x - a_i)(x - a_i)^T + c_i^{-1}$ | 4 | [0, 10] | -10.5363    |

optimization algorithms. The fixed-dimension multimodal functions are similar to high-dimensional functions, but they have a lower number of local minima.

Figure 3 illustrates a 3-D representation of a sample of the benchmark mathematical functions. As shown in that figure, some functions contain a small number of local optima while
FIGURE 3. 3-D representation of sample benchmark functions.

others have many local optima; therefore, these functions are suitable to benchmark exploration, exploitation, and convergence of optimization algorithms.

A. INTRODUCTION AND EXPERIMENTAL SETUP

Because of the random initialization of the individuals of the first population of meta-heuristic algorithms, we have run the experiments 30 times for each of the considered benchmark mathematical functions. Each run consists of 500 iterations. The population size is one of the parameters of the algorithm and has been set to include 30 individuals in our experiments. All individuals are initialized with random values. In order to validate our results, DGCO was compared with other well-known cooperative and competitive algorithms such as Particle Swarm Optimization (PSO) [20], Differential Evolution (DE) [19], Genetic Algorithm (GA) [16], Whale Optimization Algorithm (WOA) [26], and Grey Wolf Optimizer (GWO) [22]. In fact, there are many optimization algorithms in the literature. We have selected these five algorithms based on two significant factors: functionality and popularity. We selected PSO because of its well-known cooperative nature. PSO has been used to solve various optimization problems in the last decade [32]. DGCO and PSO share the same principle of cooperation among individuals to achieve their global goals. On the other hand, we selected the Genetic Algorithm (GA) and Differential Evolution (DE) as examples of evolutionary algorithms inspired by the theory of natural evolution. GA is considered as one of the most popular optimization algorithms in the literature. GA and DE have been applied to solve several real-world problems in different domains [44], [47]. Because DGCO has some evolutionary principles such as mutation and elitism, we decided to compare DGCO with popular evolutionary algorithms like GA and DE. Moreover, DGCO has been compared with state-of-the-art swarm-based algorithms such as Grey Wolf Optimizers (GWO) and Whale Optimization Algorithms (WOA). GWO and WOA were introduced in 2014 and 2016; they have been extensively used to solve several optimization problems in different domains. They are inspired by the cooperation of humpback whales and grey wolves to find their food [37], [49]. These five algorithms share some common principles with DGCO; therefore, they are suitable for comparison with our proposed algorithm.

In all experiments, we have used Python to implement the proposed optimization algorithm. For PSO, GA, DE, GWO, and WOA, we modified EvoloPy open-source [56]. The experiments have been run on a machine with the following specs: Intel i7 processor, 16 GB RAM, and Windows 10 operating system. The initial parameters of the algorithms are listed in Table 4. The obtained results have been statistically analyzed by comparing the average and relative standard deviation. Table 5 and Table 6 illustrate the obtained results for unimodal and multimodal benchmark functions. Figure 4 illustrates the convergence curves for eight benchmark functions. It should be noted that differential evolution (DE) may provide better results, similar to those in [57].
TABLE 4. The initial parameters of algorithms.

| Algorithm | Parameter          | Value          |
|-----------|--------------------|----------------|
| All       | Population size    | 30             |
|           | Number of iterations| 500            |
|           | Number of runs     | 30             |
| DGCO      | Exploration Percentage | 70         |
|           | K                  | Exponentially decreased from 2 to 0 |
|           | Mutation probability | 0.5          |
| PSO       | C1 (Cognitive Constant) | 2             |
|           | C2 (Social Constant) | 2              |
|           | W (Inertia Constant) | 0.3            |
| GA        | Parent selection   | Roulette wheel  |
|           | Crossover          | Arithmetic Recombination (probability = 1) |
|           | Mutation           | Uniform (probability = 0.5) |
| GWO       | $\tilde{d}$       | Linearly decreased from 2 to 0 |
| WOA       | $\tilde{d}$ (spiral shape) | 1          |
|           | $b$                 | 1              |
| DE        | Crossover probability | 0.5          |
|           | Mutation Factor    | 0.5            |

TABLE 5. Statistical results (mean, standard deviation) for benchmark functions (F1 – F7).

| Func, Metric | DGCO | PSO | GA | DE | WOA | GWO | PSOSCA |
|--------------|------|-----|----|----|-----|-----|--------|
| F1 Mean      | 0.00036 | 4.56E-172 | 0.000326 | 1.41E-30 | 6.59E-28 | 1.11014E-20 |
| F1 Std       | 0.000202 | 1.79E-153 | 0.000151 | 4.91E-30 | 0.000634 | 1.83289E-20 |
| F2 Mean      | 0.042144 | 3.44E-90 | 0.003702 | 1.06E-21 | 7.18E-17 | 4.09460E-11 |
| F2 Std       | 0.045421 | 6.13E-90 | 0.001040 | 2.39E-21 | 0.029014 | 5.68981E-11 |
| F3 Mean      | 70.12562 | 1.66E-127 | 35076.982 | 5.39E-07 | 0.0000033 | 2.16858E-12 |
| F3 Std       | 22.11924 | 8.64E-127 | 6024.358 | 2.93E-06 | 79.14958 | 1.03815E-11 |
| F4 Mean      | 1.086481 | 1.15E-75 | 13.492504 | 0.072581 | 0.0000006 | 8.47410E-08 |
| F4 Std       | 0.317039 | 2.45E-75 | 5.044014 | 0.39747 | 1.315088 | 1.23324E-07 |
| F5 Mean      | 0.006422 | 96.71832 | 28.372867 | 69.005638 | 27.86558 | 26.81258 | 21.97646 |
| F5 Std       | 0.000647 | 60.11559 | 0.582802 | 64.771083 | 0.763626 | 69.90499 | 0.54774 |
| F6 Mean      | 0.00088 | 0.000102 | 3.932626 | 0.000335 | 3.116266 | 0.816579 | 7.13998E-12 |
| F6 Std       | 0.000088 | 0.0000828 | 0.431755 | 0.000189 | 0.532429 | 0.000126 | 3.65884E-11 |
| F7 Mean      | 0.000473 | 0.122854 | 0.022992 | 0.047142 | 0.001425 | 0.002213 | 0.00012 |
| F7 Std       | 0.000509 | 0.044957 | 0.021966 | 0.012227 | 0.001149 | 0.100286 | 0.00010 |

and [26], if we increase the number of iterations. However, we preferred to use the same configuration (population size, number of iterations, and number of runs) as the other algorithms in our experiment to provide a fair comparison.

B. EXPLOITATION PERFORMANCE ANALYSIS

The unimodal functions (F1-F7) have only one global optimum, so they are used to benchmark exploitation of the optimization algorithms. It can be seen from Table 5 that DGCO was the most efficient optimizer for functions F1, F2, F3, F4, and F7 and had very competitive results for F5 and F6. The reason is that DGCO applies two different exploitation techniques in every iteration. The first technique is moving towards the best solution whereas the other technique is searching around the best solution. These techniques help DGCO achieve exploitation by finding better solutions around the best solution found so far. The integrated exploitation techniques are not the only reasons for the high exploitation capability of DGCO; the balance between exploitation and exploration is another important factor. Unlike most of the optimizers, DGCO starts the exploitation process at early iterations, and the number of individuals in the exploitation group increases over the course of iterations. This explains why DGCO outperformed other optimizers in most of the unimodal benchmark functions.

C. EXPLORATION PERFORMANCE ANALYSIS

The multimodal benchmark functions (F8-F23) have many local minima that increase exponentially with problem size. Therefore, they are suitable to measure exploration performance and the ability to avoid local optima. It can be seen from Table 6 that DGCO is very competitive against other optimization algorithms. The reported results in the table prove the high exploration capability of DGCO.

The reason is that DGCO applies two different exploration techniques. The first one is to search around each solution
whereas the second is the mutation that acts as a random jump to another position in the search space. Also, DGCO controls the number of individuals in different teams in order to achieve a balance between exploration and exploitation. DGCO starts with a higher number of exploring individuals at early iterations, and this number decreases over the course of iterations. In addition to that, DGCO guarantees that some individuals are exploring the search space until the last iteration. All these integrated techniques provide DGCO with the ability to highly explore the search space and find promising points that may lead to the global optimum.

**D. ABILITY TO AVOID LOCAL OPTIMA**

The multimodal benchmark functions (F8–F23) have many local optima that make the optimization process difficult and challenging. The ability of DGCO to explore the search space and avoid local optima has been confirmed by the reported results of the multimodal benchmark functions in Table 6. As shown in the table, DGCO was the best optimizer in six benchmark functions and was very competitive in other benchmark functions. Besides the two different exploration techniques, DGCO offers dynamic control of the balance between exploration and exploitation. Moreover, the mutation operator, provided by DGCO, introduces local random perturbations to some components of the exploring individuals that act as a jump to another position in the search space. Therefore, it helps in avoiding local optima. The integrated techniques guarantee a high number of exploring individuals in the early iterations.

Unlike most of the optimizers, DGCO guarantees that some individuals are exploring the search space over the entire course of iterations. Moreover, DGCO increases the number of exploring individuals if the fitness of the leaders for three consecutive iterations does not improve significantly. These techniques guarantee the ability of DGCO to avoid local optima and stagnation problems.

**E. STATISTICAL SIGNIFICANCE OF THE RESULTS**

In order to verify that the results obtained by DGCO have a statistical significance, a one-tailed t-test with a significance level equal to 50% is employed for each benchmark function. Tables 5–7 present the t-test results of comparing DGCO against other optimizers in finding the minimum of each benchmark function. Table 7 provides statistical significance test results between DGCO and each optimizer. The obtained results show that DGCO outperformed most of the surveyed optimization algorithms. DGCO outperformed other optimization algorithms in statistical significance in 86 out

---

TABLE 6. Statistical results (mean, standard deviation and p-value for t-test between DGCO and other optimizers) for benchmark functions (F8 – F23).

| Func. | Metric | DGCO | PSO | GA | DE | WOA | GWO | PSOSCA |
|-------|--------|------|-----|----|----|-----|-----|--------|
| F8    | Mean   | -7349.75 | -4841.29 | -4080.18 | -6728.93274 | -5080.76 | -6123.1 | -12569.486 |
|       | Std    | 782.604763 | 1152.814 | 551.650425 | 381.862656 | 695.7968 | 4087.44 | 2.39996E-07 |
| F9    | Mean   | 0 | 46.70423 | 0 | 151.38992 | 0 | 0.310521 | 0 |
|       | Std    | 11.62938 | 1.79E-15 | 12.041537 | 1.55E-15 | 47.35612 | 0 |
| F10   | Mean   | 4.44E-16 | 0.276015 | 7.99E-16 | 0.006564 | 7.4043 | 1.06E-13 | 2.24609E-11 |
|       | Std    | 0.50901 | 1.066E-15 | 0.002645 | 9.897572 | 0.077835 | 2.33547E-11 |
| F11   | Mean   | 0 | 0.009215 | 0 | 0.01250 | 0 | 0.004485 | 0 |
|       | Std    | 0.007724 | 1.79E-15 | 0.022357 | 1.79E-15 | 0.006659 | 0 |
| F12   | Mean   | 5.88E-05 | 0.006917 | 0.556713 | 0.034678 | 0.339676 | 0.053438 | 8.4646E-14 |
|       | Std    | 6.02E-05 | 0.026301 | 0.063582 | 0.106195 | 0.214864 | 0.020754 | 2.79106E-13 |
| F13   | Mean   | 2.23E-05 | 0.006675 | 2.13E-01 | 0.001409 | 1.889015 | 0.654464 | 0.00399 |
|       | Std    | 5.41E-05 | 0.008907 | 1.75E-01 | 0.001675 | 2.660088 | 0.004474 | 0.009284 |
| F14   | Mean   | 0.998004 | 3.627168 | 0.998004 | 0.998004 | 2.111973 | 4.042493 | 1.13027 |
|       | Std    | 9.83E-13 | 2.560828 | 1.37E-09 | 0 | 2.498594 | 4.252799 | 0.50338 |
| F15   | Mean   | 0.000367 | 0.000577 | 0.002318 | 0.000667 | 0.000572 | 0.000337 | 3.13244E-04 |
|       | Std    | 4.72E-05 | 0.000222 | 1.01E-02 | 0.000316 | 0.000324 | 0.000665 | 2.17489E-05 |
| F16   | Mean   | -1.031628 | -1.03163 | -1.031627 | -1.031628 | -1.03163 | -1.03163 | -1.0316 |
|       | Std    | 1.42E-06 | 6.25E-16 | 4.44E-06 | 0 | 4.2E-07 | -1.03163 | 4.40244E-16 |
| F17   | Mean   | 0.397897 | 0.397887 | 0.398223 | 0.397887 | 0.397914 | 0.397889 | 0.39788 |
|       | Std    | 1.60E-05 | 0 | 1.39E-03 | 0 | 2.7E-05 | 0.397887 | 3.66527E-15 |
| F18   | Mean   | 3.000014 | 3 | 3.000029 | 3 | 3.000000 | 3 | 3.000028 | 3 |
|       | Std    | 1.79E-05 | 1.33E-15 | 4.22E-05 | 0 | 4.2E-15 | 3 | 5.96540E-13 |
| F19   | Mean   | -3.862737 | -3.86278 | -3.862724 | -3.862782 | -3.85816 | -3.86263 | -3.86278 |
|       | Std    | 8.8E-05 | 2.58E-15 | 9.018E-05 | 0 | 0.002076 | -3.86278 | 8.31755E-15 |
| F20   | Mean   | -3.277191 | -3.26654 | -3.250664 | -3.258777 | -2.98105 | -3.28654 | -3.27168 |
|       | Std    | 0.058339 | 0.065016 | 0.081811 | 0.056943 | 0.376653 | -3.25056 | 0.06371 |
| F21   | Mean   | -10.098054 | -6.8651 | -6.037215 | -9.855166 | -7.04918 | -10.1514 | -10.15319 |
|       | Std    | 0.008885 | 3.019644 | 2.00E+00 | 1.340970 | 3.629551 | -9.14015 | 4.46227E-15 |
| F22   | Mean   | -10.164287 | -8.45653 | -6.768098 | -8.174817 | -8.18178 | -10.4015 | -10.40294 |
|       | Std    | 0.006955 | 3.087094 | 2.63 | 3.110998 | 3.829202 | -8.58441 | 1.80672E-15 |
| F23   | Mean   | -10.47606 | -9.95291 | -5.794591 | -10.260210 | -9.34238 | -10.55343 | -10.53640 |
|       | Std    | 0.009741 | 1.782786 | 2.64E+00 | 1.201211 | 2.414737 | -8.55899 | 4.84794E-15 |
of 115 cases. Whereas other optimization methods outperformed DGCO in significance in only 4 cases. The other optimization methods achieved better results in 9 cases but without significance.

On the other hand, DGCO achieved better results but without statistical significance in the remaining 16 cases. It is noted that in some of the 16 cases, DGCO and other optimizers could find the global optima of the function (F1, F2, F3, F9, F10, and F11) with almost the same average but with different standard deviations. This may explain why there is no statistical significance.

**F. CONVERGENCE BEHAVIOR**

In this section, we discuss the convergence behavior of DGCO. Most of the meta-heuristics algorithms tend to extensively explore the search space at early iterations by
TABLE 7. Statistical Significance test (p-value for t-test between DGCO and each optimizer) for benchmark functions (F1 – F23).

| Func. | DGCO vs PSO | DGCO vs GA | DGCO vs DE | DGCO vs WOA | DGCO vs GWO | DGCO vs PSOSCA |
|-------|-------------|------------|------------|-------------|-------------|----------------|
| F1    | 0.000493    | 0.160732   | 5.18E-16   | 0.032387    | 0.047123    | 0.001572       |
| F2    | 0.003655    | 0.079733   | 3.70E-26   | 0.075249    | 1.21E-10    | 0.000220       |
| F3    | 7.87E-15    | 0.100078   | 5.01E-40   | 4.29E-20    | 0.035678    | 0.257269       |
| F4    | 1.39E-33    | 0.041780   | 1.51E-28   | 5.76E-14    | 1.87E-06    | 0.000393       |
| F5    | 1.36E-11    | 7.50E-96   | 1.06E-07   | 4.46E-96    | 6.15E-85    | 2.06E-86       |
| F6    | 0.007435    | 1.78E-53   | 0.051794   | 7.58E-16    | 1.90E-14    | 0.000166       |
| F7    | 6.29E-05    | 5.76E-07   | 3.56E-51   | 3.68E-06    | 2.37E-13    | 0.000441       |
| F8    | 4.17E-28    | 1.37E-27   | 7.87E-05   | 1.08E-13    | 1.46E-10    | 9.91E-42       |
| F9    | 2.29E-22    | 0.160732   | 1.02E-58   | 0.160732    | 1.15E-05    | 1              |
| F10   | 0.066711    | 0.000969   | 8.06E-18   | 1.85E-16    | 5.69E-41    | 2.12E-06       |
| F11   | 2.75E-09    | 0.160732   | 0.048821   | 0.160732    | 0.024986    | 1              |
| F12   | 1.45E-05    | 9.72E-38   | 0.022639   | 2.22E-11    | 1.39E-15    | 6.02E-05       |
| F13   | 2.17E-05    | 1.89E-54   | 1.14E-10   | 1.89E-13    | 7.25E-22    | 0.022649       |
| F14   | 0.001371    | 0.000112   | 0.001371   | 0.001386    | 2.71E-01    | 0.155477       |
| F15   | 0.003187    | 0.005675   | 0.001415   | 4.72E-05    | 0.001874    | 4.82E-07       |
| F16   | 0.002621    | 0.087747   | 0.002621   | 0.002633    | 0.004447    | 1.44E-68       |
| F17   | 0.392933    | 0.089666   | 0.392933   | 0.247351    | 0.305471    | 2.70E-07       |
| F18   | 0.001726    | 0.102700   | 0.001726   | 0.484464    | 0.160733    | 7.00E-05       |
| F19   | 0.073984    | 0.093963   | 0.004036   | 0.042269    | 0.468981    | 0.009660       |
| F20   | 0.021657    | 0.455627   | 3.81E-87   | 0.039947    | 0.201872    | 0.728057       |
| F21   | 0.000244    | 1.06E-12   | 0.042387   | 0.000312    | 0.000985    | 5.41E-40       |
| F22   | 1.13E-07    | 1.27E-13   | 0.042370   | 3.27E-05    | 0.010449    | 1.75E-82       |
| F23   | 0.083927    | 7.40E-14   | 1.97E-06   | 3.40E-09    | 0.079278    | 5.98E-40       |

introducing random changes to their search agents. Then, these random changes are reduced to increase exploitation at the final iteration. This behavior guarantees that an algorithm eventually converges towards a point that could be a global optimum in the search space. Similarly, DGCO follows the same behavior by dividing the population into two teams: the exploration team and the exploitation team. In addition, DGCO controls the number of individuals in each team dynamically by starting with a higher number of individuals in the exploration team at early iterations. Then, the number of exploration individuals decreases over the course of iterations to emphasize exploitation.

Unlike most of the optimizers, DGCO guarantees that some individuals are working on the exploration team in early iterations. This explains the fast convergence behavior reported in Figure 4. The convergence behavior of DGCO is compared to PSO, GA, and GWO for some benchmark functions. Figure 4 shows that DGCO tends to converge very quickly compared to other optimizers in most of the benchmark functions. This is due to the ability of DGCO to control the balance between exploration and exploitation and starting exploitation in early iterations.

Moreover, the integrated elitism strategy improves the convergence performance of DGCO and ensures the movement of the best solution found in one iteration to the next iteration.

G. CONVERGENCE TIME

In this section, we study the execution time of the examined algorithms to find the global optima of different benchmark functions. In our experiments, we used the same configuration in Table 4. Moreover, we have added a stopping condition for all algorithms so that in each run, the algorithm stops running if it finds the global optima of a benchmark function (as described in Table 2 and Table 3). The average and the standard deviation of the time of convergence of each algorithm for each benchmark function over 30 runs are reported in Table 8. Moreover, we have reported the average number of calls to the fitness function (average function evaluations FE) in the same table. The number of function evaluations represents the computational effort required for each algorithm to find the global optima of an optimization problem. As shown in Table 9, the merit of DGCO appears in its ability to converge very quickly towards global optima.

In Table 9, DGCO is the best algorithm in terms of convergence time for most of the benchmark functions compared to other surveyed optimization algorithms. Consequently, DGCO has the lowest number of function evaluations as reported in Table 9. Being able to find the global optima very quickly with a lower number of function evaluations proves that DGCO requires less computational effort and that it is very competitive compared to other optimization algorithms.

Based on our experiments, we found that some algorithms such as PSO and DE are able to find global optima of some benchmark functions but after a higher number of iterations (around 3000-5000 iterations) which increases the convergence time and the number of function evaluations as well.
### TABLE 8. Comparison of average convergence time (avg. time) in seconds, the standard deviation for convergence time (std. time) and the average number of function evaluations (FEs) on sample benchmark functions.

| Func. | Metric   | DGCO   | PSO    | GA     | GWO    | WOA    | DE     |
|-------|----------|--------|--------|--------|--------|--------|--------|
| F1    | avg. time| 0.296  | 1.596  | 1.555  | 2.173  | 0.957  | 1.869  |
|       | std. time| 0.012  | 0.037  | 0.066  | 0.082  | 0.038  | 0.038  |
|       | avg. FEs | 5645   | 15000  | 15000  | 15000  | 15000  | 15000  |
| F2    | avg. time| 0.779  | 1.689  | 2.315  | 2.280  | 1.046  | 2.086  |
|       | std. time| 0.066  | 0.021  | 1.308  | 0.041  | 0.013  | 0.125  |
|       | avg. FEs | 13467  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F3    | avg. time| 1.964  | 4.034  | 4.043  | 4.559  | 3.334  | 5.759  |
|       | std. time| 0.166  | 0.041  | 0.086  | 0.098  | 0.044  | 0.486  |
|       | avg. FEs | 5587   | 15000  | 15000  | 15000  | 15000  | 15000  |
| F4    | avg. time| 0.704  | 1.569  | 1.514  | 2.120  | 0.900  | 1.792  |
|       | std. time| 0.060  | 0.079  | 0.024  | 0.031  | 0.008  | 0.029  |
|       | avg. FEs | 13617  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F6    | avg. time| 0.821  | 1.640  | 1.590  | 2.195  | 0.983  | 1.907  |
|       | std. time| 0.007  | 0.041  | 0.034  | 0.025  | 0.024  | 0.018  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F8    | avg. time| 0.829  | 1.706  | 1.387  | 2.168  | 1.003  | 2.228  |
|       | std. time| 0.029  | 0.010  | 0.011  | 0.011  | 0.046  | 0.253  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F9    | avg. time| 0.836  | 1.735  | 1.625  | 2.255  | 1.034  | 2.230  |
|       | std. time| 0.008  | 0.088  | 0.116  | 0.027  | 0.012  | 0.213  |
|       | avg. FEs | 389    | 15000  | 1605   | 15000  | 5768400| 1530   |
| F10   | avg. time| 1.083  | 1.935  | 1.841  | 2.434  | 1.227  | 2.275  |
|       | std. time| 0.036  | 0.024  | 0.015  | 0.034  | 0.011  | 0.078  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F11   | avg. time| 0.047  | 2.001  | 0.228  | 1.385  | 1.290  | 2.968  |
|       | std. time| 0.009  | 0.069  | 0.017  | 0.718  | 0.060  | 0.706  |
|       | avg. FEs | 406    | 15000  | 1839   | 15000  | 15000  | 15000  |
| F12   | avg. time| 1.600  | 2.431  | 2.394  | 3.005  | 1.766  | 3.396  |
|       | std. time| 0.034  | 0.016  | 0.013  | 0.097  | 0.079  | 0.460  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F13   | avg. time| 1.518  | 2.346  | 2.322  | 2.848  | 1.698  | 3.176  |
|       | std. time| 0.029  | 0.009  | 0.016  | 0.042  | 0.063  | 0.398  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F15   | avg. time| 0.781  | 0.789  | 1.230  | 0.815  | 0.681  | 1.479  |
|       | std. time| 0.030  | 0.100  | 0.008  | 0.005  | 0.008  | 0.031  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F17   | avg. time| 0.356  | 0.276  | 0.777  | 0.316  | 0.272  | 0.961  |
|       | std. time| 0.014  | 0.008  | 0.011  | 0.011  | 0.049  | 0.026  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F21   | avg. time| 3.302  | 3.305  | 3.603  | 3.299  | 3.360  | 5.107  |
|       | std. time| 0.028  | 0.039  | 0.020  | 0.036  | 0.247  | 0.094  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
| F23   | avg. time| 6.708  | 6.839  | 7.112  | 6.711  | 6.997  | 9.812  |
|       | std. time| 0.142  | 0.214  | 0.023  | 0.067  | 0.421  | 0.091  |
|       | avg. FEs | 15000  | 15000  | 15000  | 15000  | 15000  | 15000  |
Moreover, DGCO is equipped with the elitism technique, which guarantees that the best solution found in one iteration will not be lost in the next iteration. This helps improve the convergence process of the algorithm.

Based on the reported results in Tables 5-9 and Figure 4, DGCO proved to be a promising competitor to existing optimization algorithms in terms of convergence behavior, convergence time, and computational effort expected from an optimization algorithm.

V. DGCO FOR CONSTRAINED ENGINEERING DESIGN PROBLEMS

In order to examine the performance of solving real-world constrained engineering design optimization problems, we have applied DGCO to three well-known engineering design problems: pressure vessel design, tension/compression spring design, and welded beam design. The general constrained optimization problem along with equality, inequality, and lower bound and upper bound constraints can be expressed as follows:

Minimize \( f(x) \)

Subject to the constraints:

\[
g_j(x) = 0, \quad j = 1, 2, \ldots, p
\]

\[
g_j(x) \leq 0, \quad j = 1, 2, \ldots, m
\]

\[
l_i \leq x_i \leq u_i, \quad i = 1, 2, \ldots, n
\]  \( (8) \)

where \( f(x) \) is the objective function, \( g_j(x) = 0 \) are the equality constraints, \( p \) is the total number of equality constraints, \( g_j(x) \geq 0 \) are the inequality constraints, \( m \) is the total number of inequality constraints, \( l_i \) and \( u_i \) are the lower bound and the upper bound of \( x_i \) respectively, and \( n \) is the number of design variables (dimension of the solution) \([58]\).

Due to the fact that these constrained optimization problems have several equality and inequality constraints, we have employed scalar penalty functions \([59]\) to model and handle these constraints. By using the penalty functions, the unconstrained optimization problem is converted to a constrained optimization problem. The new objective function can be described as follows:

\[
\min \varphi_k(x) = f(x) + p_k \sum_{i \in I} g(c_i(x)) \quad (9)
\]

where \( g(c_i(x)) = \max(0, c_i(x))^2 \) is the penalty function, \( c_i(x) \) are the constraints of the problem, \( \varphi_k(x) \) is the new objective function, \( f(x) \) is the objective function of the problem, and \( p_k \) is the penalty coefficient.

By applying this penalty rule, we guarantee that any individual that violates the constraints will be assigned a high objective function value. For these constrained optimization problems, we used similar penalty functions of those adopted in \([60]\) and set the penalty coefficient to \( p_k = 10^{15} \).

![FIGURE 5. Pressure vessel design problem.](image)

A. PRESSURE VESSEL DESIGN PROBLEM

In the pressure vessel design problem, the main goal is to minimize the total cost of material, forming, and welding of the cylindrical pressure vessel shown in Figure 5. There are four continuous design variables and four inequality constraints. The problem can be formulated as follows:

Minimize \( f(T_s, T_h, R, L) = 0.6224T_s RL + 1.7781T_h R^2 \\
+ 3.1661T_s^2 L + 19.84T_h^2 L \)

Subject to the constraints:

\[
g_1 = -T_s + 0.0193R \leq 0
\]

\[
g_2 = -T_h + 0.0095R \leq 0
\]

\[
g_3 = -\pi R^2 L - \frac{4}{3} \pi R^3 + 1, 296, 000 \leq 0
\]

\[
g_4 = L - 240 \leq 0
\]  \( (10) \)

where Variables range as follows:

\[ 0 \leq T_s \leq 99, \quad 0 \leq T_h \leq 99, \quad 10 \leq R \leq 200, \quad 10 \leq L \leq 200 \]
TABLE 10. Comparison of DGCO results with literature for pressure vessel design problem.

| Algorithm | Best       | Average | Standard deviation | Function evaluations |
|-----------|------------|---------|--------------------|----------------------|
| DGCO      | 5893.80371722 | 6083.0895194 | 161.529487603 | 6220                |
| PSO [61]  | 6061.0777   | 6531.10  | 154.3716           | 14790               |
| GWO [22]  | 5993.5876   | 6283.236 | 401.2634           | 7600                |
| GSA [26]  | 8538.8359   | 8932.95  | 683.5475           | 7110                |

where $T_s$, $T_h$, $R$ and $L$ are the thickness of the shell, the thickness of the head, the inner radius, and the length of the cylindrical section of the vessel respectively.

This problem has been solved by many researchers using different techniques such as GA [60], PSO [61], GWO [22], and mathematical methods like the augmented lagrangian multiplier (ALM) [62]. Table 9 shows the performance of DGCO on this problem. This table shows the optimal values for the design variables corresponding to the best solution obtained by DGCO, GA [60], PSO [61], GWO [22], WOA [26], ALM [62], and GSA [26]. As can be seen, DGCO outperforms other optimization algorithms, and it is able to find the best feasible optimal design for the pressure vessel design problem.

Table 10 compares the statistical results obtained by DGCO and some of the other algorithms when solving the pressure vessel design problem over 30 independent runs. We have used 20 individuals over 500 iterations to solve this problem. It may be observed in this table that DGCO has the best average compared to the other methods. In addition, DGCO outperforms other methods in finding the best optimal design with the least number of fitness evaluations. The integrated exploration and exploitation techniques in DGCO assist in finding the best optimal design variables for this problem. Moreover, being able to find the optimal values with the least number of fitness evaluations is another proof of the fast convergence behavior of DGCO.

FIGURE 6. Tension/compression spring design problem.

B. TENSION/COMPRESSION SPRING DESIGN PROBLEM

The objective of this problem is to minimize the weight of the spring (as shown in Figure 6) subject to constraints on deflection, stress, surge frequency, and geometry [48]. The problems involve three continuous variables and four nonlinear inequality constraints. The design variables are the wire diameter ($w$), the mean coil diameter ($d$), and the length or number of coils ($L$). The mathematical model of the spring design problem can be expressed as follows:

Minimize $f(w, d, L) = (L + 2) w^2 d$

Subject to the constraints:

\[
g_1 = 1 - \frac{d^3L}{71785 w^4} \leq 0
\]

\[
g_2 = \frac{1}{12566(6w^3 - w^4)} + \frac{1}{5108w^2} - 1 \leq 0
\]

\[
g_3 = 1 - \frac{140.45w}{d^2L} \leq 0
\]

\[
g_4 = \frac{2(w + d)}{3} - 1 \leq 0
\]

where Variables range as follows:

\[
0.05 \leq w \leq 2.0,
\]

\[
0.25 \leq d \leq 1.3,
\]

\[
2.0 \leq L \leq 15
\]

Table 11 illustrates the performance of DGCO on this problem. This table shows the optimal values for the design variables and the optimal cost obtained by DGCO, GA [60], PSO [61], GSA [26], DE [48], GWO [22], and WOA [26]. It may be observed in this table that DGCO outperforms the other methods, and it is able to find the optimal design for the tension/compression spring design problem.

The statistical results of DGCO and some of the other algorithms when solving the tension/compression spring design problem are presented in Table 12. To solve this problem, we have used 20 individuals, a maximum of 500 iterations, and 20 independent runs. According to the results in this table, DGCO was the second-best optimizer in terms of average. In addition, DGCO was able to find the best optimal design for the problem with the least number of function evaluations. The results prove the ability of DGCO to highly explore the search space and quickly converge towards the optimal goal.

C. WELDED BEAM DESIGN PROBLEM

This problem is considered one of the standard engineering optimization problems [63]. The problem involves the following four design variables: the width $w$ and the length $L$ of the welded area and the depth $h$ and the thickness $d$ of the main beam. The objective of this problem is to minimize the fabrication cost of the welded beam (as shown in Figure 7) subject to the following constraints: shear stress $\tau$, bending stress $\sigma$, buckling load $P$, and maximum end deflection $\delta$. 

The problem can be formulated as follows:

\[
\begin{align*}
&\text{Minimize} \\
&f(w, L, d, h) = 1.10471w^2L + 0.04811dh(14.0 + L)
\end{align*}
\]

Subject to the constraints:

\[
\begin{align*}
g_1 &= w - h \leq 0 \\
g_2 &= \delta - 0.25 \leq 0 \\
g_3 &= \tau - 13,600 \leq 0 \\
g_4 &= \sigma - 30,000 \leq 0 \\
g_5 &= 0.125 - w \leq 0 \\
g_6 &= 6000 - P \leq 0 \\
g_7 &= 0.10471w^2 + 0.04811dh(14 + L) - 0.5 \leq 0 
\end{align*}
\]

where

\[
\begin{align*}
\sigma &= \frac{504,000}{hd^2} \\
Q &= 6000(14 + \frac{L}{2}) \\
D &= \frac{1}{2} \sqrt{L^2 + (w + d)^2}
\end{align*}
\]

\[
J = \sqrt{2} wL \left( \frac{L^2}{6} + \frac{(w + d)^2}{2} \right)
\]

\[
\delta = \frac{65856}{30000hd^3}
\]

\[
\tau = \sqrt{\alpha^2 + \left( \frac{\alpha \beta L}{D} \right) + \beta^2}
\]

\[
\alpha = \frac{6000}{\sqrt{2} wL}
\]

\[
\beta = \frac{QD}{J}
\]

\[
P = 0.61432 \times 10^6 \frac{dh^3}{6} \left( 1 - \frac{d\sqrt{30/48}}{28} \right)
\]
TABLE 13. Comparison of the best solution for the welded beam design problem.

| Algorithm | W   | L    | d    | h    | Optimal Cost |
|-----------|-----|------|------|------|--------------|
| GA [60]   | 0.205986 | 3.471328 | 9.020224 | 0.206480 | 1.728226    |
| GA [65]   | 0.208800 | 3.420500 | 8.997500 | 0.210000 | 1.748309    |
| PSO [61]  | 0.202369 | 3.544214 | 9.048210 | 0.205723 | 1.728024    |
| GSA [26]  | 0.182129 | 3.856979 | 10.00000 | 0.20376  | 1.879952    |
| RO [64]   | 0.203687 | 3.528467 | 9.004233 | 0.207241 | 1.735344    |
| WOA [26]  | 0.205396 | 3.484293 | 9.037426 | 0.206276 | 1.730499    |
| DGCO      | 0.20550286 | 3.47118533 | 9.05051895 | 0.20571302 | 1.726866    |

TABLE 14. Comparison of DGCO statistical results with literature for the welded beam design problem.

| Algorithm | Best Value | Average Value | Standard Deviation | Function Evaluation |
|-----------|------------|---------------|--------------------|---------------------|
| DGCO      | 1.726866   | 1.8163        | 0.0972224106123    | 7440                |
| PSO [61]  | 1.728024   | 1.7422        | 0.01275            | 13770               |
| GSA [26]  | 1.879952   | 3.5761        | 1.2874             | 10750               |
| WOA [26]  | 1.730499   | 1.7320        | 0.0226             | 9900                |

FIGURE 8. Speed reducer design problem.

The design variables are: face width \( x_1 \), module of the teeth \( x_2 \), number of teeth on the pinion \( x_3 \), length of the first shaft between bearings \( x_4 \), length of the second shaft between bearings \( x_5 \), diameter of the first shaft \( x_6 \), and diameter of the second shaft \( x_7 \). The problem can be formulated as follows:

Minimize

\[
f(x) = 0.7854x_1x_2^2 \\
\times \left( 3.3333x_2^2 + 14.9334x_3 - 43.0934 \right) \\
- 1.508x_1 \left( x_6^2 + x_7^2 \right) + 7.4777 \left( x_6^3 + x_7^3 \right) \\
+ 0.7854 \left( x_4x_6^2 + x_5x_7^2 \right)
\]

Subject to the constraints:

\[
g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0
\]
\[
g_2(x) = \frac{397.5}{x_1x_2^2x_3} - 1 \leq 0
\]
\[
g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^3} - 1 \leq 0
\]
\[
g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^3} - 1 \leq 0
\]
\[
g_5(x) = \frac{x_2x_3}{40} - 1 \leq 0
\]
\[
g_6(x) = \frac{5x_2}{x_1} - 1 \leq 0
\]
\[
g_7(x) = \frac{x_6}{12x_2} - 1 \leq 0
\]
\[
g_8(x) = \frac{x_4}{1.5x_6 + 1.9} - 1 \leq 0
\]
\[
g_9(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0
\]
\[
g_{10}(x) = \frac{1}{110x_5^3} \sqrt{\left( \frac{745x_4}{x_2x_3} \right)^2 + 16.9 \times 10^6 - 1}
\]
\[
g_{11}(x) = \frac{1}{85x_5^3} \sqrt{\left( \frac{745x_5}{x_2x_3} \right)^2 + 157.5 \times 10^6 - 1}
\]

where Variables range as follows:

\[
2.6 \leq x_1 \leq 3.6 \\
0.7 \leq x_2 \leq 0.8 \\
17 \leq x_3 \leq 28 \\
7.3 \leq x_4 \leq 8.3 \\
7.8 \leq x_5 \leq 8.3 \\
2.9 \leq x_6 \leq 3.9 \\
5.0 \leq x_7 \leq 5.0
\]

Table 15 shows the performance of DGCO on the speed reducer design problem. This table shows the optimal values for the design variables corresponding to the best solution obtained by DGCO, GA, PSO, GWO, and WOA. As can be seen, DGCO outperforms other optimization algorithms, and it is able to find the best feasible optimal design for speed reducer design problem.
TABLE 15. Comparison of the best solution for the speed reducer design problem.

| Algorithm | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | Optimal Cost |
|-----------|-------|-------|-------|-------|-------|-------|-------------|
| GA        | 3.5024564 | 0.7  | 17.0  | 7.3   | 7.8   | 3.35176812 | 5.28922439  | 2999.3257   |
| PSO       | 3.50846793 | 0.7  | 17.0  | 7.3   | 7.8   | 3.3630089  | 5.28726908  | 3003.3185   |
| GWO       | 3.50530051 | 0.7  | 17.0  | 7.36966809 | 7.81677503 | 3.3510112 | 5.2906983  | 3002.1709   |
| WOA       | 3.51839084 | 0.7  | 17.0  | 7.3   | 7.9968466 | 3.37798453 | 5.28765156  | 3015.0628   |
| DGCO      | 3.50018314 | 0.7  | 17.0  | 7.40536402 | 7.8022425  | 3.35156015 | 5.28803278  | 2998.60038  |

Table 16 compares the statistical results obtained by DGCO and some of the other methods when solving the speed reducer design problem over 30 independent runs. We have used 20 individuals over 500 iterations to solve this problem. It may be observed in this table that DGCO has the best average compared to other methods. Moreover, DGCO outperforms other methods in finding the best optimal design with the least number of fitness evaluations. The complexity of this constrained engineering problem and the superiority of DGCO prove that DGCO can handle constrained engineering problems and reflect the high exploration and exploitation capabilities of DGCO.

E. HIMMELBLAU’S NONLINEAR OPTIMIZATION PROBLEM

This problem is one of the well-known benchmark problems for optimization algorithms. It was proposed by Himmelblau. The problem involves five design variables and six nonlinear inequality constraints. The problem is formally described as follows:

Minimize $f(x) = 5.3578547x_1^2 + 0.8356891x_1x_5 + 37.29329x_1 - 40792.141$

Subject to the constraints:

$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5$  
$g_2(x) = 80.51249 + 0.0071317x_2x_5 + 0.002995x_1x_2 - 0.0002181x_3^2$  
$g_3(x) = 9.30961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4$  
$0 \leq g_1(x) \leq 92,$  
$90 \leq g_2(x) \leq 110,$  
$20 \leq g_3(x) \leq 25$  
(14)

where Variables range as follows:

$78 \leq x_1 \leq 102,$  
$33 \leq x_2 \leq 45,$  
$27 \leq x_3, x_4, x_5 \leq 45$

Table 17 illustrates the performance of DGCO and some of the other methods on this problem. This table includes the optimal design variable corresponding to the optimal cost obtained by DGCO, GA, PSO, GWO, and WOA. As may be seen in this table, DGCO outperforms the other methods, and it is able to find the optimal design for Himmelblau’s Nonlinear optimization problem.

The statistical results obtained by DGCO and the other methods are illustrated in Table 18. We have used 20 individuals over 500 iteration and 20 independent runs. As may be seen in this table, DGCO has the best average for this problem. In addition, DGCO was able to find the optimal variable with the least function evaluations which confirms the fast convergence behavior of DGCO. These results prove that DGCO can be used to solve constrained problems efficiently.

VI. SENSITIVITY ANALYSIS OF DGCO PARAMETERS

In this section, we investigate the sensitivity analysis of DGCO with respect to its parameters. DGCO has five parameters: population size, number of iterations, mutation rate, exploration percentage, and k vector. Table 19 lists the parameters of DGCO along with their descriptions and default values. Those parameters control the performance of the algorithm in solving optimization problems. Any single change in one parameter may affect the optimization process. Hence, a sensitivity analysis of these parameters is conducted in order to collect information that will help improve the algorithm in future runs. In order to perform a sensitivity analysis of DGCO’s parameters, we selected the multimodal benchmark function (F23), as described in Table 3.
TABLE 17. Comparison of the best solution for Himmelblau’s Nonlinear Optimization Problem.

| Algorithm | $x_1$ | $x_2$ | $x_3$ | $x_4$ | Optimal Cost |
|-----------|-------|-------|-------|-------|--------------|
| GA        | 78.0  | 33.0  | 27.1253641 | 44.94007203 | 44.86239533 | -310167381   |
| PSO       | 78.0  | 33.0  | 27.09235129 | 45.0  | 44.92880907 | -310219989   |
| GWO       | 78.0  | 33.0  | 27.07302577 | 44.97379974 | 44.97379974 | -310207148   |
| WOA       | 78.21013303 | 33.08890243 | 33.08890243 | 45.0  | 44.7278826 | -30990.6451 |
| DGCO      | 78.0  | 33.0  | 27.07346179 | 44.9833164 | 44.97289873 | -310246069   |

TABLE 18. Comparison of DGCO statistical results with literature for Himmelblau’s Nonlinear Optimization Problem.

| Algorithm | Best      | Average   | Standard deviation | Function evaluation |
|-----------|-----------|-----------|--------------------|---------------------|
| DGCO      | -31024.6069 | -31015.2802 | 19.944            | 1920                |
| PSO       | -31021.9989 | -31013.5328 | 3.5285            | 1360                |
| GA        | -31016.7381 | -30524.3625 | 427.7116          | 4180                |
| GWO       | -31020.7148 | -31013.3387 | 3.1077            | 3440                |
| WOA       | -30990.6451 | -30165.9736 | 398.396510902     | 2680                |

TABLE 19. Description of DGCO’s parameters.

| Parameter Name | Description | Default Value |
|----------------|-------------|---------------|
| Population Size | The number of search agents in the population. | 25 |
| Iterations Count | The number of iterations that the algorithm will run. | 250 |
| Mutation Rate | Controls diversity of the population. | 0.5 |
| Exploration Percentage | Controls the initial number of search agents in the exploration group. DGCO decreases | 50 |
| $K$ | The value of $k$ decreases from the initial value to zero over the course of 1 iterations. | |

A. ONE-AT-A-TIME SENSITIVITY ANALYSIS

In order to perform the sensitivity analysis, we have applied the One-at-a-time (OAT) sensitivity measure [66]. OAT is considered as one of the simplest approaches to sensitivity analysis. OAT consists of measuring the performance of the algorithm while changing one parameter at a time and keeping the other parameters constant. Table 20 and Table 21 list the observed changes in time and the fitness values of DGCO while changing the values of different parameters. As shown in Table 20 and Table 21, we have selected 20 different values in the interval of each parameter by adding 5% of the length of that interval to obtain a new value for evaluation. For each of those values, the algorithm was run 10 times and the averages for both time and fitness have been reported on the table. Each parameter involved 200 different runs of DGCO.

Figure 9 lists the curves of convergence time and fitness for each parameter. As shown in the figure, each parameter has two curves: one for convergence time and another for fitness. Iterations count and population size proved to be the most effective parameters on the convergence time of the algorithm. This is verified by the fact that increasing the number of individuals or iterations will increase the number of calls to the objective function, which will increase the computational cost and the time for convergence. On the other hand, higher values of vector $K$ slightly decrease convergence time. Moreover, exploration percentages greater than 20 have shown a good effect on the convergence time of the algorithm. A mutation rate with values lower than 0.6 has shown a clear effect on the convergence time. For the fitness of the algorithm, increasing iterations count or vector $K$ enhances the fitness of the algorithm. In addition, population size values greater than 50 search agents enhance the fitness of the algorithm. Population size values greater than 50 also have a good effect on the performance of the algorithm.

B. REGRESSION ANALYSIS

For a further study on how parameters can explain the variations of the performance of the algorithm, a regression analysis has been performed. Regression analysis is adequate when we need to predict the value of a dependent variable (performance of the algorithm) upon the value of an independent variable (parameter).

Table 22 lists the results of regression analysis for the parameters of DGCO, convergence time, and fitness. The value of R Square indicates how much of the total variation of time or fitness can be explained by the values of the parameter. As listed in Table 22, iterations count and population size have the highest value of R Square for convergence time which means that they can explain any variation in
convergence time very well. On the other hand, the vector $k$ value can explain 57.59% of the variation in the convergence time of the algorithm. Moreover, the exploration percentage and the mutation rate can explain 37.28% and 24.81% of the variation in convergence time respectively. In Table 22, significance F with values lower than 0.05 indicates that

| Mutation Rate | Exp. Percentage | Population Size | Iterations Count | K-Vector |
|---------------|-----------------|-----------------|------------------|---------|
| values        | values | time | values | time | values | time | values | time | values | time |
| 0.050         | 4.918  | 5.000 | 4.822  | 10    | 1.563  | 10    | 0.903  | 0.100 | 4.936  |
| 0.100         | 4.710  | 10.000 | 5.282  | 20    | 3.088  | 20    | 1.809  | 0.200 | 4.893  |
| 0.150         | 4.583  | 15.000 | 5.007  | 30    | 4.660  | 30    | 2.711  | 0.300 | 4.827  |
| 0.200         | 5.068  | 20.000 | 4.814  | 40    | 6.498  | 40    | 3.603  | 0.400 | 4.811  |
| 0.250         | 4.816  | 25.000 | 4.613  | 50    | 7.845  | 50    | 4.500  | 0.500 | 4.867  |
| 0.300         | 4.679  | 30.000 | 4.614  | 60    | 9.269  | 60    | 5.413  | 0.600 | 4.824  |
| 0.350         | 4.636  | 35.000 | 4.609  | 70    | 10.818 | 70    | 6.351  | 0.700 | 4.854  |
| 0.400         | 4.625  | 40.000 | 4.646  | 80    | 12.382 | 80    | 8.214  | 0.800 | 4.785  |
| 0.450         | 4.602  | 45.000 | 4.604  | 90    | 13.889 | 90    | 9.028  | 0.900 | 4.850  |
| 0.500         | 4.606  | 50.000 | 4.604  | 100   | 15.446 | 100   | 11.791 | 1.000 | 4.869  |
| 0.550         | 4.593  | 55.000 | 4.624  | 110   | 17.004 | 110   | 12.688 | 1.100 | 4.842  |
| 0.600         | 4.602  | 60.000 | 4.632  | 120   | 19.527 | 120   | 13.560 | 1.200 | 4.773  |
| 0.650         | 4.694  | 65.000 | 4.623  | 130   | 20.857 | 130   | 14.576 | 1.300 | 4.843  |
| 0.700         | 7.906  | 70.000 | 4.599  | 140   | 21.775 | 140   | 15.363 | 1.400 | 4.842  |
| 0.750         | 7.837  | 75.000 | 4.606  | 150   | 23.147 | 150   | 17.457 | 1.500 | 4.791  |
| 0.800         | 7.647  | 80.000 | 4.613  | 160   | 24.738 | 160   | 18.429 | 1.600 | 4.778  |
| 0.850         | 5.844  | 85.000 | 4.607  | 170   | 27.332 | 170   | 18.648 | 1.700 | 4.783  |
| 0.900         | 6.135  | 90.000 | 4.625  | 180   | 32.502 | 180   | 19.437 | 1.800 | 4.776  |
| 0.950         | 4.905  | 95.000 | 4.623  | 190   | 37.067 | 190   | 20.980 | 1.900 | 4.760  |
| 1.000         | 5.359  | 95.000 | 4.607  | 200   | 36.642 | 200   | 21.832 | 2.000 | 4.751  |
the regression model is statistically significant to predict the performance of the algorithm.

**C. STATISTICAL SIGNIFICANCE OF THE RESULTS**

In order to determine whether there is any statistical difference between the means of different observations listed in Table 20 and Table 21, we used a one-way analysis of variance (ANOVA). We conducted two ANOVA tests for both convergence time and fitness values while changing the parameters of DGCO. Table 23 lists the results of ANOVA test for the convergence time and the minimum fitness of DGCO. In Table 23, we can see that p-values are less than 0.05 and F is greater than F-critical. Therefore, there is a statistically significant difference between the means of the five groups of convergence time observed by changing the values of each parameter. Moreover, there is a statistically significant difference between the means of the five groups of minimum fitness observed by changing the values of each parameter.

ANOVA test does not provide any information about which groups have a statistical significance. Therefore, a post hoc test is conducted between every two groups. For this purpose, we have employed a one-tailed T-Test at 0.05 significance level. Table 24 lists the results of the T-Test conducted for each pair of parameters based on values observed for convergence time and minimum fitness of DGCO. As listed in the table, p-values less than 0.05 indicate that there a statistically significant difference between groups. For convergence time, the p-value is greater than 0.05 for the T-Test between exploration percentage and mutation rate. This indicates that...
TABLE 24. Results of one-tailed T-Test (0.05 significance) for observed time and fitness for each pair of parameters.

| Parameter Pairs                      | T-Test (P-value) for time | T-Test (p-value) for Min. fitness |
|--------------------------------------|---------------------------|-----------------------------------|
| Mutation Rate, Exploration Percentage| 0.008126971               | 3.41456E-09                       |
| Mutation Rate, Population Size       | 7.49074E-06               | 3.70149E-14                       |
| Mutation Rate, Iterations Count      | 0.00021051                | 0.464616675                       |
| Mutation Rate, K                     | 0.025183094               | 0.283278774                       |
| Exploration Percentage, Population Size | 2.94406E-06            | 0.000194321                       |
| Exploration Percentage, Iterations Count | 5.1121E-05            | 6.8888E-07                        |
| Exploration Percentage, K            | 0.001068148               | 1.1383E-05                        |
| Population Size, Iterations Count    | 0.021958447               | 4.43838E-11                       |
| Population Size, K                   | 3.49752E-06               | 8.40752E-09                       |
| Iterations Count, K                  | 6.61723E-05               | 0.281205251                       |

there is no statistically significant difference between their effects on the convergence time. In addition, mutation rate and iterations count have no statistically significant difference in minimum fitness. Mutation rate and vector k also have no statistical significance in terms of minimum fitness. Similarly, iterations count and vector k have no statistical significance for minimum fitness.

D. DISCUSSION AND RANKING OF PARAMETERS

According to Table 22, the parameters of DGCO can be ranked based on their effect on the fitness values as the following: Vector K, Iterations Count, Mutation Rate, Exploration Percentage, and Population Size. Moreover, DGCO’s parameters can be ranked based on their effect on the convergence time as the following: Iterations Count, Population Size, Vector K, Exploration Percentage, and Mutation Rate. The mutation rate has the least effect on the convergence time of DGCO. It is noted that DGCO uses mutation as one of two techniques for exploring the search space. This means that 50% of the exploration group is using mutation. Moreover, the main role of mutation is to avoid stagnation into local optima. The convergence speed depends highly on the exploitation group. All of these reasons explain why the mutation rate has the least effect on the convergence of the algorithm.

As a summary, the convergence time of DGCO is highly responsive to population size and iterations count. Moreover, the convergence time of DGCO is very responsive to the values of exploration percentage greater than 25. The convergence time has a notable response for K values greater than one and mutation rate values greater than 0.7. The fitness value is strongly affected by population sizes greater than 70. Moreover, vector K has shown a good impact on the fitness for values greater than one. The fitness of the algorithm is slightly affected by the iterations count, mutation rate and exploration percentage.

VII. CONCLUSION

In this paper, we proposed a novel meta-heuristic optimization algorithm called Dynamic Group-based Cooperative Optimization Algorithm (DGCO). It is inspired by the way individuals of a group cooperate to achieve a global goal. DGCO divides the individuals into two groups: exploration group and exploitation group. The exploration group applies two different techniques (Search around individuals and Mutation) to highly explore the search space for promising areas. On the other hand, the exploitation group applies two techniques (Move towards the leader and Search around the leader) to find better solutions from good ones. DGCO allows dynamic control of the number of individuals in each group to guarantee the balance between exploration and exploitation.

A comparative study has been conducted on twenty-three standard benchmark mathematical functions to evaluate the performance of the proposed algorithm in finding the optimal points of each function and to study its convergence behavior. The obtained results showed that DGCO is very competitive with state-of-the-art meta-heuristic algorithms. DGCO has a fast convergence behavior due to its high exploration and exploitation capabilities and the ability to avoid local optima. Moreover, DGCO has been tested for solving constrained engineering design optimization problems (pressure vessel design, tension/compression spring design, and welded beam design). The results showed that DGCO is able to solve real-world constrained optimization problems. A sensitivity analysis of the parameters of DGCO has been conducted to study the impact of the parameters on the performance of DGCO.

For future work, we are planning to deploy parallel and multi-objective versions of DGCO. Moreover, DGCO will be applied to solve further optimization problems.
REFERENCES

[1] E. L. Grossmann, Ed., Global Optimization in Engineering Design, vol. 9. Springer, 2013, doi: 10.1007/978-1-4757-5331-8.
[2] E.-S.-M. El-Kenawy, M. M. Eid, M. Saber, and A. Ibrahim, “MbGWO-Global Optimization in Engineering Design,” vol. 2018, pp. 1–17, Sep. 2018, doi: 10.1155/2018/9147568.
[3] W.-Y. Lin, “A novel 3D fruit fly optimization algorithm and its applications in economics,” Neurocomput., vol. 27, no. 5, pp. 1391–1413, Jul. 2016, doi: 10.1016/j.neucom.2015.01.1498-2.
[4] M. Nouri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, “An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem,” J. Intel1. Manuf., vol. 29, no. 3, pp. 603–615, Mar. 2018, doi: 10.1007/s10845-017-1265-9.
[5] X. Wang, T.-M. Choi, H. Liu, and X. Yue, “A novel hybrid ant colony optimization algorithm for emergency transportation problems during post-disaster scenarios,” IEEE Trans. Syst., Man, Cybern., vol. 48, no. 4, pp. 545–556, Apr. 2018, doi: 10.1109/TSMC.2018.2806440.
[6] Y. Li, J. Wang, D. Zhao, G. Li, and C. Chen, “A two-stage approach for combined heat and power economic emission dispatch: Combining multi-objective optimization with integrated decision making,” Energy, vol. 162, pp. 237–254, Nov. 2018, doi: 10.1016/j.energy.2017.07.200.
[7] R. Al-Hajj and A. Assi, “Estimating solar irradiance using genetic programming technique and meteorological records,” AIMS Energy, vol. 5, no. 5, pp. 798–813, 2017, doi: 10.3934/energy.2017.5.798.
[8] R. Al-Hajj, A. Assi, and F. Batch, “An evolutionary computing approach for estimating global solar radiation,” in Proc. IEEE Int. Conf. Renew. Energy Res. Appl. (ICRERA), Nov. 2016, pp. 285–290, doi: 10.1109/ICRERA.2016.784855.
[9] E.-G. Talbi, Metaheuristics: From Design to Implementation, vol. 74, Hoboken, NJ, USA: Wiley, 2009.
[10] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, “Metaheuristic research: A comprehensive survey,” Artif. Intell. Rev., vol. 52, no. 4, pp. 2191–2233, Dec. 2019, doi: 10.1007/s10462-017-9605-x.
[11] M. Laguna, “Tabu search,” in Handbook of Heuristics, vols. 1–2, Boston, MA, USA: Springer, 2018, pp. 741–758.
[12] B. Sunam and P. Kumar, “A survey of simulated annealing as a tool for single and multiobjective optimization,” J. Oper. Res. Soc., vol. 57, no. 10, pp. 1143–1160, Oct. 2006, doi: 10.1057/palgrave.jors.2602068.
[13] M. Dorigo and T. Stützle, “The Ant colony optimization metaheuristic: Algorithms, applications, and advances,” in Handbook of Metaheuristics, Boston, MA, USA: Kluwer Academic, 2003, pp. 250–285.
