International Journal of Mathematical, Engineering and Management Sciences
Vol. 11, No. 4, 1590-1621, 2026
https://doi.org/10.33889/IJMEMS.2026.11.4.065

Multi-Objective Social Group Optimization with Dynamic Fitness Function and Crowding Distance Elimination

Ghazwan Alsoufi
Department of Operations Research and Intelligent Techniques,
College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq.
E-mail: ghazwan.alsoufi@uomosul.edu.iq

Manal Abdulkareem Zeidan
Department of Operations Research and Intelligent Techniques,
College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq.
Corresponding author: manal.alabaji@uomosul.edu.iq

Niam Abdulmunim Al-Thanoon
Department of Operations Research and Intelligent Techniques,
College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq.
E-mail: niam.munim@uomosul.edu.iq

Xinan Yang
School of Mathematics, Statistics and Actuarial Science,
University of Essex, United Kingdom.
E-mail: xyangk@essex.ac.uk

(Received on January 31, 2026; Revised on March 30, 2026 & May 21, 2026 & June 8, 2026; Accepted on June 17, 2026)

Abstract
Many real-world optimization problems involve conflicting objectives that need to be minimized to reduce cost and/or maximized to increase profit. In this study, a multi-objective social group optimization (MOSGO) is proposed and implemented to solve multi-objective problems and find approximated solutions to the optimal Pareto front. A new mechanism, the dynamic fitness function, is introduced and integrated with non-dominated sorting and crowding distance elimination strategies to enhance the quality of the non-dominated solutions. The dynamic fitness function is designed to select the best solution for each objective at each iteration. Non-dominated sorting is used to dismiss weak solutions, and crowding distance elimination is deployed to achieve the best solution diversity. The suggested algorithm is compared with four competitive algorithms: the multi-objective artificial hummingbird algorithm (MOAHA), the multi-objective particle swarm optimization (MOPSO), the multi-objective ant lion optimizer (MOALO), and the non-dominated sorting genetic algorithm-II (NSGA-II). Computational simulations are performed on well-studied ZDT benchmark test functions. Comprehensive comparisons are carried out regarding convergence, diversity, and solution distribution. Experiment results show that the proposed MOSGO provides, in most problems, significantly better convergence near the true Pareto front, with improved diversity and spread of solutions, compared to other multi-objective algorithms.

Keywords- Multi-objective optimization, Social group optimization, Convergence and diversity, Dynamic fitness function, Non-dominated sorting, Crowding distance elimination.

1. Introduction and Literature Review

The development of computer science has improved operations research (OR) techniques that assist decision-makers in optimizing their problems and achieving optimal solutions. In practice, most problems confronting decision-makers involve finding solutions that reduce costs and/or increase productivity/profitability. In the past, one of the most familiar ways to deal with multi-objective problems was to assign weights to objectives so that objectives with high importance were given higher weights and

objectives with low importance were given lower weights. Nevertheless, the choice of weights plays a vital role in the optimization process. Over the last few decades, with the growth of the complexity of real-world problems and the need to deal with conflicting aims, multi-objective optimization (MOO) has been implemented to solve these problems and achieve reliable solutions. MOO, introduced by Vilfredo Pareto, is an efficient technique for generating solution sets that define the best tradeoff amongst conflicting objectives while satisfying several criteria.

The simple concept of multi-objective problems is that there are n conflicting objectives that should be optimized to obtain a non-dominated solution set called the Pareto front (PF). Over the years, many approaches, known as multi-objective algorithms (MOAs), have been introduced and developed in this field. The key procedure of MOA is to find the trade-off points among conflicting objectives that allow the decision-makers to achieve approximated solutions to the optimal Pareto front and select the best solution depending on their experience. In recent years, MOAs have become a popular choice to find solutions to the complex multi-objective problems (Coello et al., 2021). With the development of MOAs, more complex multi-objective problems become tractable in less computation time and effort. Many multi-objective algorithms have been introduced and implemented to generate non-dominated Pareto front solutions for multi-objective problems, like the vector-evaluated genetic algorithm (VEGA) (Schaffer, 1985), strength Pareto evolutionary algorithm (SPEA) (Zitzler and Thiele, 1998), Pareto envelope-based selection algorithm (PESA) (Corne et al., 2000), multi-objective particle swarm optimization (MOPSO) (Coello et al., 2004), multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li, 2007), multi-objective artificial bee colony (MOABC) (Akbari et al., 2012), multi-objective cuckoo search (MOCS) (Yang and Deb, 2013), multi-objective gray wolf optimizer (MOGWO) (Mirjalili et al., 2016), non-dominated sorting whole optimization algorithm (NSWOA) (Jangir and Jangir, 2017), multi-objective ant colony optimization (MOACO) (Ning et al., 2018), multi-objective artificial sheep algorithm (MOASA) (Lai et al., 2019), multi-objective moth swarm algorithm (MOMSA) (Sharifi et al., 2021) and multi-objective butterfly optimization algorithm (MOBOA) (Xiao et al., 2024) among others. Regarding their promising results, these algorithms have received more attention in the literature. Moreover, Coello (2006) provided a general overview of the evolutionary algorithms for multi-objective problems (MOPs) and addressed the most developed algorithms and their applications. Methodological problems concerning the usage of multi-objective evolutionary algorithms are investigated. Sharma and Kumar (2022) illustrated the multi-objective algorithms and their variations, along with their advantages and disadvantages, and covered representative algorithms in each category and the applications of various MOPs in engineering domains. The authors urged that MOPs face open challenges and discussed important characteristics of MOPs that will assist new researchers in implementing these algorithms. For further interest in MOPs, please see (Fonseca and Fleming, 1995; Zitzler, 1999; Marler and Arora, 2004).

Articles that are closely related to this study include: Deb et al. (2002), who suggested the non-dominated sorting genetic algorithm-II (NSGA-II). In addition, a selection operator generates a mating pool by mixing the parent and offspring populations and selecting the best (based on fitness and spread solutions). Coello et al. (2004) suggested a strategy in which Pareto dominance is included in particle swarm optimization (PSO). This allows the heuristic to tackle issues with multi-objective functions. An archive of particles is created, which is then used by other particles to direct their flight. Additionally, a specific mutation operator is applied to improve the exploratory capabilities of the proposed algorithm. Mirjalili et al. (2017) introduced the multi-objective ant lion optimizer (MOALO). They prepared an external archive to hold the non-dominated Pareto front solutions. Then, solutions are chosen from this archive using a roulette wheel method that uses the coverage of solutions as ant lions to direct ants to promising parts of the multi-objective search space. Liu and Chen (2019) proposed a crowding distance elimination (CDE) method to improve the non-dominated solution diversity. The crowding distance for the non-dominated solutions is calculated,

and the solution with the smallest crowding distance is removed. Then, the crowding distance of the remaining solutions is recalculated, and the solution with the smallest crowding distance is also skipped. The above process is repeated until the number of non-dominated solutions matches the required solutions. Naik et al. (2021) presented non-dominated sorting social group optimization (NSSGO) as a multi-objective algorithm. Non-dominated sorting and crowding distance strategies are implemented to refresh the solutions. Zhao et al. (2022) presented a multi-objective artificial hummingbird algorithm (MOAHA) for solving multi-objective problems. A repository to save Pareto solutions is used to maintain this repository and effectively preserve population variety, and a dynamic elimination-based crowding distance (DECD) is introduced. In addition, a non-dominated sorting approach is combined with MOAHA to form a solution update mechanism.

When comparing the outputs of multiple objective algorithms and selecting acceptable indicators to perform those comparisons, there is no perfect agreement on which metrics to utilize. Many measures have been developed to assess the quality of Pareto front approximations generated by these algorithms. Riquelme et al. (2015) presented a review and analysis of 54 multi-objective indicators in the literature, discussing the tendency, usage, advantages, and disadvantages of the most mentioned ones to provide researchers with adequate information when selecting indicators. They determined that the most commonly used indicator is hypervolume, followed by generational distance, epsilon indicator, and inverted generational distance. Audet et al. (2021) submitted a review of 63 performance indicators classified into four groupings based on their properties: spread, cardinality, convergence, and dispersion. Additionally, the application of these metrics is introduced. For more information about the indicators of MOPs, please see (Falcón-Cardona and Coello, 2020; Shang et al., 2021).

From another angle, Satapathy and Naik (2016) suggested and successfully implemented social group optimization (SGO) as a human social behavior-based algorithm. As a single-objective algorithm, SGO outperforms other metaheuristic methods. SGO's performance and characteristics make it a good contender for solving multi-objective problems. Therefore, this article inherits the solution framework of SGO and extends the solution process to make it implementable on multi-objective problems to obtain the Pareto front. To facilitate this, a new mechanism, the dynamic fitness function, is introduced and integrated with non-dominated sorting and crowding distance elimination strategies to improve the non-dominated solutions. The main contributions of the work are:

- Creates a novel multi-objective algorithm called multi-objective social group optimization (MOSGO).
- Proposes a dynamic fitness function scheme to find the best solution for each objective at each iteration.
- Applies a non-dominated solution update mechanism based on front ranking to improve convergence.
- Implements a crowding distance elimination strategy to create a diverse Pareto front.
- Evaluates MOSGO's efficiency against three cutting-edge algorithms in five benchmark multi-objective test functions.

The differences between the proposed algorithm and the most recent and relevant work of Naik et al. (2021) on NSSGO are presented as follows:

- In MOSGO, the dynamic fitness function for each objective function at each iteration is proposed and implemented to select the best solution (X_{best}), whereas in NSSGO, a solution located at the first rank (first Pareto front) is considered the best solution (X_{best}). If more than one person has the same rank, a person with a greater crowding distance value is considered. The dynamic fitness function allows the algorithm to achieve better and faster convergence to the new solutions.

- In MOSGO, the crowding distance elimination is implemented, whereas in NSSGO, the ordinary crowding distance is applied. As is known, crowding distance elimination gives better diversity to Pareto front solutions than ordinary crowding distance.
- In this article, we use ZDT problems (described in Section 4.1) in numerical tests. ZDT problems are standard, difficult test problems used to evaluate the performance of multi-objective algorithms, compared to the relatively easier ones (e.g., CEC 2009) used in Naik et al. (2021) for the test of NSSGO.
- In MOSGO, we use a much higher self-introspection parameter (0.8), compared to that used in NSSGO (0.2). This means that in NSSGO, the $Xnew$ solution (offspring) produced in the improving phase inherits just a little knowledge from the $Xold$ solution, which makes the learning less effective.

This paper is divided into the following sections: introduction and the literature review, corresponding to the multi-objective problems, are presented in Section 1. A problem description is given in Section 2. The social group optimization and its development to cope with multi-objective problems are presented in Section 3. The proposed algorithm is validated by computational experiments presented in Section 4. Finally, Section 5 covers the conclusions and several future directions.

2. Problem Description

2.1 Multi-Objective Problems (MOPs)

In multi-objective problems (MOPs), two or more conflicting objective functions need to be optimized so that the cost is minimized or the profit is maximized, subject to some constraints according to the type of problem in Zitzler et al. (2000). The formula of multi-objective optimization can be defined as follows:

$$Minimize\ f(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_n(\vec{x})]$$

$$Subject\ to:$$

$$g_i(x) \le 0, \quad i = 1, 2, \dots, p \tag{1}$$

$$h_j(x) = 0, \quad j = 1, 2, \dots, q$$

$$L_i \le x_i \le U_i$$

where, $f_i(\vec{x})$ is the i^{th} objective function, g_i refers to the i^{th} inequality constraint, and h_i refers to the i^{th} equality constraint. L_i and U_i indicate the lower bound and upper bound of the i^{th} variable, respectively. p and q represent the number of inequality and equality constraints, respectively.

MOPs are more difficult and complex than single-objective problems (SOPs). The MOP difficulty stems from several factors, including determining the best solution in evolutionary algorithms, as most of these algorithms require the best solution at each iteration. Determining the best solution in SOP is straightforward. In contrast, in MOP it becomes more complex because there are multiple objectives, and the selected solution may be suitable for one objective but not for others. In addition, the solution of SOP is just one point, whereas for MOP, the solution should be represented by a set (Pareto front) that includes all the non-dominated solutions. Also, the conflict among the objective functions in solving MOPs makes it difficult to simultaneously satisfy optimization criteria for all objective functions.

2.2 Non-Dominated Sorting (NDS)

Deb et al. (2002) introduced a fast non-dominated sorting method. It plays a significant role in determining the quality of the solutions and then improving solution convergence. The non-dominated sorting mechanism ranks candidate solutions based on Pareto dominance. Let (x_1) and (x_2) be two solutions in the

search space. Then (x_1) is said to dominate (x_2) if:
$[\forall i \in 1, \ldots, M, f_i(x_1) \leq f_i(x_2), \text{and } \exists\, j \text{ such that } f_j(x_1) < f_j(x_2)]$.

where, M denoted the number of objective functions. The elitist selection strategy of NDS is achieved by iteratively maintaining the non-dominated solutions in the generation. This ensures that any Pareto optimal solution found is not rejected in subsequent iterations and thus encourages convergence towards the true Pareto front. The concept of non-dominated solutions in the multi-objective algorithms means that if there are two solutions A and B, for a problem with two objectives, i and j. Solution A dominates solution B if and only if the fitness functions i and j for the solution A are less than the fitness functions i and j for solution B for a minimization problem, and vice versa. Whereas solution A does not dominate the solution B if the fitness function i for the solution A is less than the fitness function i for the solution B, and the fitness function j for the solution A is greater than the fitness function j for the solution B. Using this concept, solutions are sorted by ranks. The first set of non-dominated solutions is given the 1st rank, the second set of non-dominated solutions is given the 2nd rank, and so on.

Figure 1 illustrates the non-dominated solution sets, featuring three sorted Pareto fronts and their corresponding ranking mechanism. The green curve with five solutions is located in the 1st rank, and all these solutions are non-dominated, i.e., no solution is better than another. The 2nd rank is assigned to the blue curve, which has five solutions, all of which are dominated by solutions in the 1st rank but do not dominate each other. The red curve is ranked 3rd, with four solutions. The solutions in the 1st rank (green curve) dominate all solutions in the 2nd and 3rd ranks, whereas solutions in the 2nd rank dominate all solutions in the 3rd.

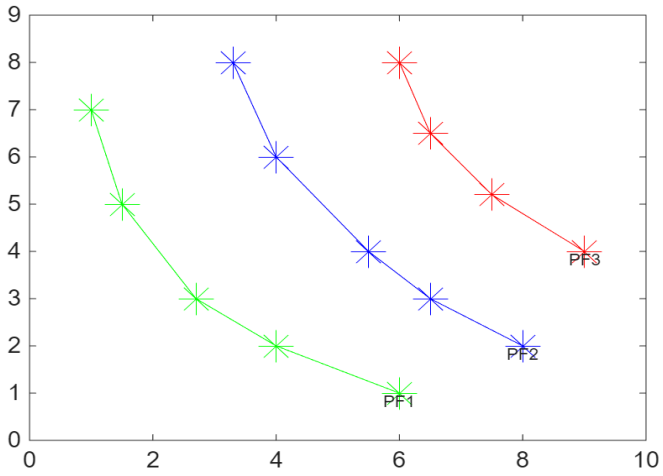

Figure 1. Ranking non-dominated solutions.

2.3 Crowding Distance (CD)

Most multi-objective algorithms require a specific number of solutions to match the population size. Algorithms will select solutions from the 1st rank (PF1) before moving to the 2nd rank (PF2) and so on to fill and update the population at each iteration. However, the number of PFs selected might contain more solutions than are required. Here, the crowding distance will be consulted to meet the desired number of solutions.

Crowding distance (CD) values indicate the density of non-dominated solutions within a solution's neighborhood. CD is frequently implemented in MOAs to maintain the solution diversity (Deb et al., 2002) and achieve the required number of solutions. In the crowding distance process, solutions in each PF are sorted into increasing order according to the fitness value of the considered objective. Assigning indices (i) to all solutions in the given PF after sorting, density for the corresponding objective j is calculated using Equation (2), in relation to the difference between the fitness values of objective j of solutions x_{i-1} and x_{i+1} divided by the difference between the maximum and minimum fitness values of objective j observed in this PF. The crowding distance of the solution x_i, as shown in Equation (3), is the summation of all objective-specific densities. Solutions that have the smallest density from their neighbourhood solutions in the same PF are those to be removed.

$$d_j(x_i) = \frac{|f_j(x_{i-1}) - f_j(x_{i+1})|}{f_{j,max} - f_{j,min}} \quad (2)$$

$$CD(x_i) = \sum_{j=1}^{n} d_j(x_i) \quad (3)$$

where, j = 1,2,...,n denotes the index of the objective and i denotes the index of the non-dominated solution. Solutions that have the largest crowding distance values are considered desirable. Similarly, solutions that have the smallest crowding distance values are considered undesirable. This is because a smaller distance means that there is more than one solution having similar characteristics, so removing one in this case will not affect the final Pareto front. To achieve greater diversity of Pareto front solutions, the solutions having the largest crowding distance values are selected as candidates to remain in the population. The extreme solutions (first and last) are assumed to have infinite crowding distance and will certainly be copied to the population.

Figures 2 and **3** demonstrate the crowding distance procedure for a problem with two objective functions. **Figure 2** shows all non-dominated solutions in the chosen PF, which contains 7 points with their objective values shown in parentheses right after the point indices. Their crowding distances were calculated using Equations (2) and (3) and are displayed next to the point as cd values. Suppose the population needs only 5 solutions to match the required number of solutions. Therefore, the two solutions that have the smallest crowding distance should be removed. These two solutions are x_4 with a crowding distance equal to 0.47 and x_3 with a crowding distance equal to 0.55. The remaining non-dominated solutions, and their refined crowding distance values, are displayed in **Figure 3**.

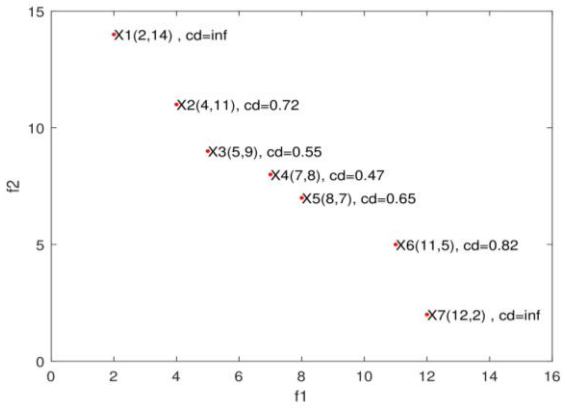

Figure 2. Non-dominated solutions.

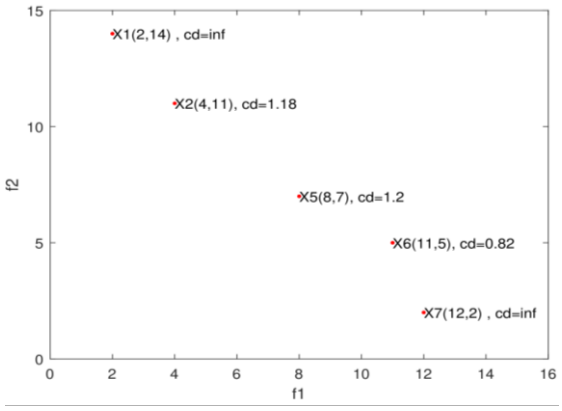

Figure 3. After crowding distance elimination.

3. Proposed Method

In this section, social group optimization is presented and extended to multi-objective social group optimization.

3.1 Social Group Optimization (SGO)

Satapathy and Naik (2016) presented social group optimization (SGO) addressing single-objective problems. The concept of human social behavior is used to address complicated optimization issues. The SGO process can be broken into two stages. The first stage is the improving phase, in which each person increases their knowledge level through the influence of the best person (solution) in the group. The best person in the group is the one with the most knowledge and problem-solving abilities. The second stage is the inquiring phase, in which each member attempts to gain new knowledge by mutually interacting with another member of the group and the best member of the group simultaneously. The mathematical formula for (single objective) SGO is described as follows:

Let $X_i, i = 1,2, \dots, N$ represents the number of solutions in the social group. Each solution X_i is defined by $X_{ij} = (x_{i1}, x_{i2}, \dots\ x_{iD})$, where D represents the number of features assigned to a solution in the social group that determines the dimensions of a solution. Let $f(X_i), i = 1,2, \dots, N$ be their corresponding fitness values.

3.1.1 Improving Phase

The person (or solution) with the highest knowledge level in each social group represents the best person (X_{best}). This person can transmit his/her knowledge to all people (solutions) in the group to help them develop their knowledge level. Therefore, the (X_{best}) at generation (iter) can be computed as $X_{best} = min\{f(X_i),\ i = 1,2, \dots, N\}$ for a minimization problem. The improving phase is where each solution improves its knowledge by interacting with the best solution in the group (X_{best}). Here, knowledge refers to the character's change with the influence of the best solution's character. The updates of each solution in the improving phase can be described as follows:

Algorithm 1: Pseudo-code of the improving phase in SGO

```
for i = 1:N do
    for j=1:D do
        Let X_best,j be the value of the j-th element of the current best solution
        Xnew_i,j = C * Xold_i,j + r * (X_best,j − Xold_i,j)
    end for
end for
```

where, C is known as the self-introspection parameter, whose value is set such that $0 < C < 1$, r is a random number drawn from a uniform distribution, i.e. $r \sim U(0,1)$. If $Xnew$ gives a better fitness value than that of old, then $Xnew$ is accepted and $Xold$ will be removed from the population. Otherwise, keep the $Xold$ solution in the population.

3.1.2 Inquiring Phase

In the inquiring phase, each member (solution) of the social group tries to increase his/her knowledge level by simultaneously interacting with the best solution (X_{best}) and with another random solution in the group. As is known, (X_{best}) is the best solution that has the greatest knowledge level in the group and will affect a person by teaching him/her new knowledge. A solution will also obtain new knowledge if the selected random solution from the group has more knowledge than him/her. Let (X_{best}) be the updated values at the end of the improving phase, the inquiring phase process is expressed as in **Algorithm 2,** where r_1 and

r_2 are two independent random values drawn from a uniform distribution, i.e. $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$. If $Xnew$ gives a better fitness value than that of $Xold$ then $Xnew$ is accepted, and $Xold$ will be removed from the population. Otherwise, keep the $Xold$ solution in the population. The stopping criterion is the maximum number of iterations.

Algorithm 2: Pseudo-code of the inquiring phase in SGO

for i = 1:N **do**
  Randomly select one person X_r, where $r \neq i$
  for j=1:D **do**
    Let $X_{best,j}$ be the value of the j^{th} element of the current best solution
    if $f(X_i) < f(X_r)$ **then**
      $Xnew_{i,j} = Xold_{i,j} + r_1 * (Xold_{i,j} - X_{r,j}) + r_2 * (X_{best,j} - Xold_{i,j})$
    else
      $Xnew_{i,j} = Xold_{i,j} + r_1 * (X_{r,j} - Xold_{i,j}) + r_2 * (X_{best,j} - Xold_{i,j})$
    end if
  end for
end for

3.2 Multi-Objective Social Group Optimization (MOSGO)

Extending the ideas of SGO as presented above, this subsection is dedicated to designing a multi-objective social group optimization (MOSGO) approach to generate solutions close to the optimal Pareto front. MOSGO consists of five main components: the improving phase, the inquiring phase, the Dominance Fitness Function (DFF), Non-Dominated Sorting (NDS), and Crowding Distance Estimation (CDE). These components are described in the following subsections.

3.2.1 Dynamic Fitness Function (DFF)

In single-objective problems, it is straightforward to determine the best solution, which is the one with the smallest fitness for minimization problems. The final solution will be just one best solution, not a set of solutions. This is due to the ability to compare any two fitness functions and select the smallest of them. In contrast, multi-objective optimization problems (MOPs) have multiple conflicting objectives, which makes selecting a suitable solution for all the objective functions a complicated process. As there is more than one objective, the comparison will be impossible. The solution might have the smallest fitness for one objective, while a large fitness for other objectives. For this reason, a solution set (non-dominated solutions) will be generated as a solution in solving MOPs. Many factors could influence and change the solutions of MOPs using optimization algorithms. One of the most important factors in solving multi-objective optimization problems is selecting the best solution for any algorithm. Since most multi-objective algorithms (MOAs) rely on a guiding solution (X_{best}) to direct the evolutionary process, the selection of an inappropriate (X_{best}) can adversely affect convergence across the entire set of objectives. Consequently, the mechanism used to identify or construct this guiding solution plays a critical role in ensuring balanced search behavior and drives the convergence of solutions for all objective functions.

The Dynamic Fitness Function (DFF) represents an innovative strategy, which is proposed and applied to cope with multiple conflicting objectives for identifying a suitable (X_{best}) by alternating the selection criterion at each iteration. Rather than attempting to optimize all objectives simultaneously, which often results in a stagnant non-dominated set, the DFF focuses on one objective at a time in a round-robin manner. For instance, if the problem has two objectives, i.e., f_1 and f_2, then the proposed Dynamic Fitness Function (DFF) works as:

- Iteration 1: X_{best} is assigned to the solution with the minimum fitness for f_1, regardless of its f_2 value.
- Iteration 2: X_{best} is assigned to the solution with the minimum fitness for f_2, regardless of its f_1 value.

This process continues iteratively until the maximum number of generations is reached.

While non-dominated sorting and crowding distance are well-established independently in multi-objective optimization, their specific sequential integration with the dynamic fitness function as proposed here has not, to our knowledge, been previously reported in the literature. Focusing on one objective at a time works well with the integration of non-dominated sorting (NDS), as the improvement in one fitness for a solution will prevent this solution from being dominated by other solutions and therefore prevent it from being dropped in the next iteration, according to the concept of the NDS scheme. Indeed, most improved solutions obtained will be candidates for the first Pareto front because they dominate other solutions. In the next iteration, these solutions will be improved again for another objective, which, eventually, leads to convergence to the PF. In a bi-objective problem, each objective has a 50% chance to guide evolution during an iteration. In a tri-objective problem, each objective contributes with a 33% chance. Integrated with the improving and inquiring procedures proposed below, the method could effectively explore the entire PF without losing convergence.

3.2.2 Improving Phase

The improving phase depends on the best person (X_{best}) who has the highest knowledge level in the social group, which is simply the solution that has the smallest fitness in solving single objective problems. As mentioned in Section 3.2.1, determining the best solution is complicated when solving MOPs because of the conflict among the objective functions. In MOSGO, the dynamic fitness function to select (X_{best}) will be applied at each iteration to choose the objective to consider. This means that the (X_{best}) at an iteration is the solution that gives the minimum fitness for the considered objective when solving a minimization problem. The improving phase methodology for MOSGO is summarized in **Algorithm 3** below:

Algorithm 3: Pseudo-code of the improving phase in MOSGO

for i = 1:N **do**
  for j=1:D **do**
    Let $X_{best,j}$ be the value of the j^{th} element of the solution that gives the smallest fitness value of the considered objective
    $Xnew_{i,j} = C * Xold_{i,j} + r * \left(X_{best,j} - Xold_{i,j}\right)$
    if $Xnew_{i,j} \in [L, U]$ where L and U denote the lower and upper bounds of the domain of variable j **then**
      Accept $Xnew_{i,j}$
    else
      $Xnew_{i,j} = L + (U - L) * \alpha$
    end if
  end for
end for

where, C is known as the self-introspection parameter, whose value is set such that $0 < C < 1$, r and α are random numbers drawn from a uniform distribution, i.e., $r \sim U(0,1)$ and $\alpha \sim U(0,1)$. Compared to the original SGO improving process (**Algorithm 1**), an additional if condition is added to **Algorithm 3** to ensure the newly generated point is in the domain of the function definition, to be applicable to the testing problems with domains. When the newly generated solution is observed outside the domain, a random value is assigned to $Xnew_{i,j}$. This enables full exploration of the feasible region and avoids the process from

getting stuck at the boundary.

Using **Algorithm 3**, all newly generated $Xnew$ solutions are added to the population (old solutions) to extend the population set. Non-dominated sorting method (see Section 3.2.4) will then be applied to sort these solutions into PFs, and the first PF (rank1) will be selected to continue the next phase, which is the inquiring phase. Details on population management can be found in Section 3.2.4.

3.2.3 Inquiring Phase

In the inquiring phase, each social group member (solution) tries to earn new knowledge by interacting with the best person (X_{best}) and with a random person in the group for inquiring knowledge. In the inquiring phase in SGO, as noticed in **Algorithm 2**, there is a comparison between the fitness of the considered person and a random person, to choose one of the two formulas to generate the new solution. In MOSGO, there is more than one objective, and the comparison will be impossible. Here, just one equation will be implemented, and the X_{best} will be selected using the DFF scheme to overcome the difficulty of selecting X_{best}. The inquiring phase process is expressed as given below:

Algorithm 4: Pseudo-code of the inquiring phase in MOSGO

Let X_{best} be the solution that gives the smallest fitness value of the considered objective, updated after the improving phase
for i = 1:N **do**
  Randomly select one person X_r, where $r \neq i$
  for j=1:D **do**
    Let $X_{best,j}$ be the j^{th} element of X_{best}
    $Xnew_{i,j} = Xold_{i,j} + r_1 * (Xold_{i,j} - X_{r,j}) + r_2 * (X_{best,j} - Xold_{i,j})$
    if $Xnew_{i,j} \in [L, U]$ where L and U denote the lower and upper bounds of the domain of variable j **then**
      Accept $Xnew_{i,j}$
    else
      $Xnew_{i,j} = L + (U - L) * \alpha$
    end if
  end for
end for

where, r_1, r_2 and α are independent random sequences, $r_1 \sim U(0,1)$, $r_2 \sim U(0,1)$ and $\alpha \sim U(0,1)$. Compared to the original SGO inquiring process (**Algorithm 2**), an additional if condition is added to **Algorithm 4** to ensure the newly generated point is in the domain of the function definition, to be applicable to the testing problems with domains. When the newly generated solution is observed outside the domain, a random value is assigned to $Xnew_{i,j}$. This enables full exploration of the feasible region and avoids the process from getting stuck at the boundary.

Similar to the improving phase, after applying **Algorithm 4**, all newly generated $Xnew$ solutions are added to the population, on which non-dominated sorting (see Section 3.2.4) is applied to sort these solutions into PFs. The first PF (rank1) will be selected to continue to the next iteration, which starts again from the improving phase. For the full algorithm structure of MOSGO, please see **Figure 6**.

3.2.4 Non-Dominated Sorting (NDS)

In SGO, after each iteration of the improving and inquiring phases, if the fitness value of the new solution is better than that of the old solution, the old solution is replaced by the new solution. However, in MOSGO, the conflict among the objective functions complicates the solution update process. As we are searching for the Pareto front instead of a single solution, the updating scheme must be carefully redesigned using non-dominated sorting (NDS).

To update the population set, old and new solutions resulting from the improving (or inquiring) phases are combined and sorted using the NDS approach. In this process, solutions are assigned to ranked PFs, where solutions having a higher dominance hierarchy are assigned smaller rank values. In this study, only solutions that are in PF1 will be carried over to the next iteration. As a result, there are two situations for updating a certain solution:

1) When the number of non-dominated solutions is greater than the population size, the crowding distance elimination mechanism is implemented to remove the undesirable solutions until the required number of solutions is satisfied.

2) When the number of non-dominated solutions is less than the required number of solutions, repeat the non-dominated solutions NS times, where NS is equal to:

$$NS = \left\lceil \frac{\text{the population size}}{\text{the number of nondominated solutions}} \right\rceil \quad (4)$$

Then, the number of non-dominated solutions will be greater than the population size. Hence, the crowding distance elimination mechanism is implemented to achieve the required number of solutions. It is important to note that the duplication of solutions reduces the genetic diversity in the parent population temporarily, but the possible search bias is compensated in the subsequent exploration phase. The algorithm still has the ability to explore the solution space globally, because the same parent solutions are generating very diverse and different offspring due to the integration of random numbers (ranging between 0 and 1) and random partner selection in the inquiry phase.

For example, assume that the population size equals 100 and the number of non-dominated solutions is equal to 60. Here, to satisfy the population size, Equation (4) will be applied to obtain: $NS = \lceil (100/60) \rceil = 2$. By repeating the non-dominated solutions two times, the number of non-dominated solutions will equal 120. The crowding distance elimination strategy will then be implemented to obtain the exact number required for the population size. This approach, essentially, creates multiple copies of the same solution in the PF1. However, in the next iteration with **Algorithms 3** and **4**, the newly generated $Xnew$ will be different with the randomly selected r values. Considering solutions in PF1 dominate solutions in all other PFs, this process provides higher chances for current good solutions to generate offspring.

3.2.5 Crowding Distance Elimination (CDE)

To effectively maintain the solution diversity and update the population during the iterations of the optimization procedure, the non-parameter crowding distance elimination (CDE) method proposed by Liu and Chen (2019) is employed. The idea of the CDE method is to generate a Pareto front solution with high diversity. Unlike the ordinary crowding distance approach, which calculates all CD values in one go and removes all excessive solutions with the smallest CD values, this approach removes one solution (with the smallest CD value) at a time and recalculates the remaining solutions' CD values before removing the next. This iterative process guarantees that the correct solutions are removed, considering the CD value changes that occurred during the removal process.

Figures 4 and **5** show the sorting change in crowding distance values for the problem with two objectives by applying the crowding distance elimination method. The original non-dominated solutions are demonstrated in **Figure 2**, including seven solutions with their corresponding crowding distance values. To satisfy the requirements of the population, just five solutions are needed. CDE begins by computing the values of crowding distance for all the solutions except the first and last fitness function values, which are assumed to be infinity. The smallest crowding distance value is the one for the solution x_4, equal to 0.47, which makes x_4 the first one to be removed. Applying the ordinary crowding distance of **Figure 2**, the solution x_3 with a crowding distance value equal to 0.55 should be removed next, which results in the final set of solutions as seen in **Figure 3**. However, if we recalculate the crowding distance values after eliminating x_4, as shown in **Figure 4**, we can see the CD values for the remaining nodes are different from those of **Figure 2**. In this case, the solution x_2 becomes the one to be removed instead of x_3 with the updated crowding distance evaluation. This results in the final PF as seen in **Figure 5**. In short, CD values could have changed with the elimination of a node, so we should update the CD values before continuing the removal process.

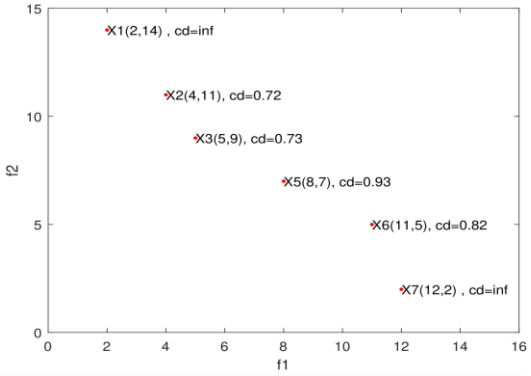

Figure 4. After removing x_4.

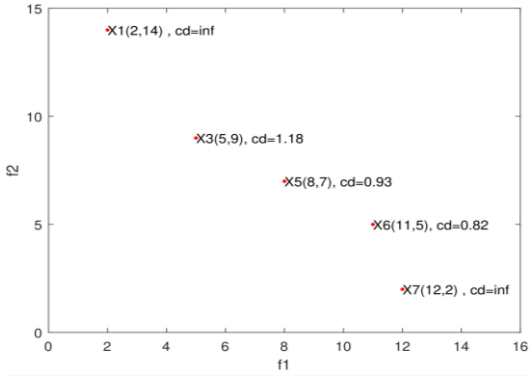

Figure 5. After removing x_4 and x_2.

The complete steps of the proposed MOSGO are illustrated in a flowchart, which is presented in **Figure 6**.

Although the proposed MOSGO algorithm is primarily evaluated empirically, its design is supported by several theoretical considerations. First, the social-group update mechanism, which includes the improving and inquiring phases, provides complementary search behaviors. The improving phase enhances exploitation by guiding solutions toward the current best candidate. In contrast, the inquiring phase introduces exploration through stochastic interactions among individuals. Second, the components of the proposed MOSGO framework establish a dynamic balance between convergence and diversity, which is a necessary condition for approximating a well-distributed Pareto front. The Dynamic Fitness Function (DFF) introduces a cyclic objective-guided selection mechanism by alternating the focus among objective functions. This mechanism can be interpreted as a stochastic decomposition strategy, where each objective is periodically emphasized, thereby promoting balanced convergence toward different regions of the Pareto front. Furthermore, the integration of non-dominated sorting (NDS) ensures that the population evolves toward the Pareto-optimal set by preserving non-dominated solutions across iterations, which aligns with the dominance-based convergence principles widely adopted in multi-objective optimization. The use of crowding distance elimination (CDE), particularly with iterative recalculation, maintains population diversity and prevents excessive concentration in specific regions of the solution space.

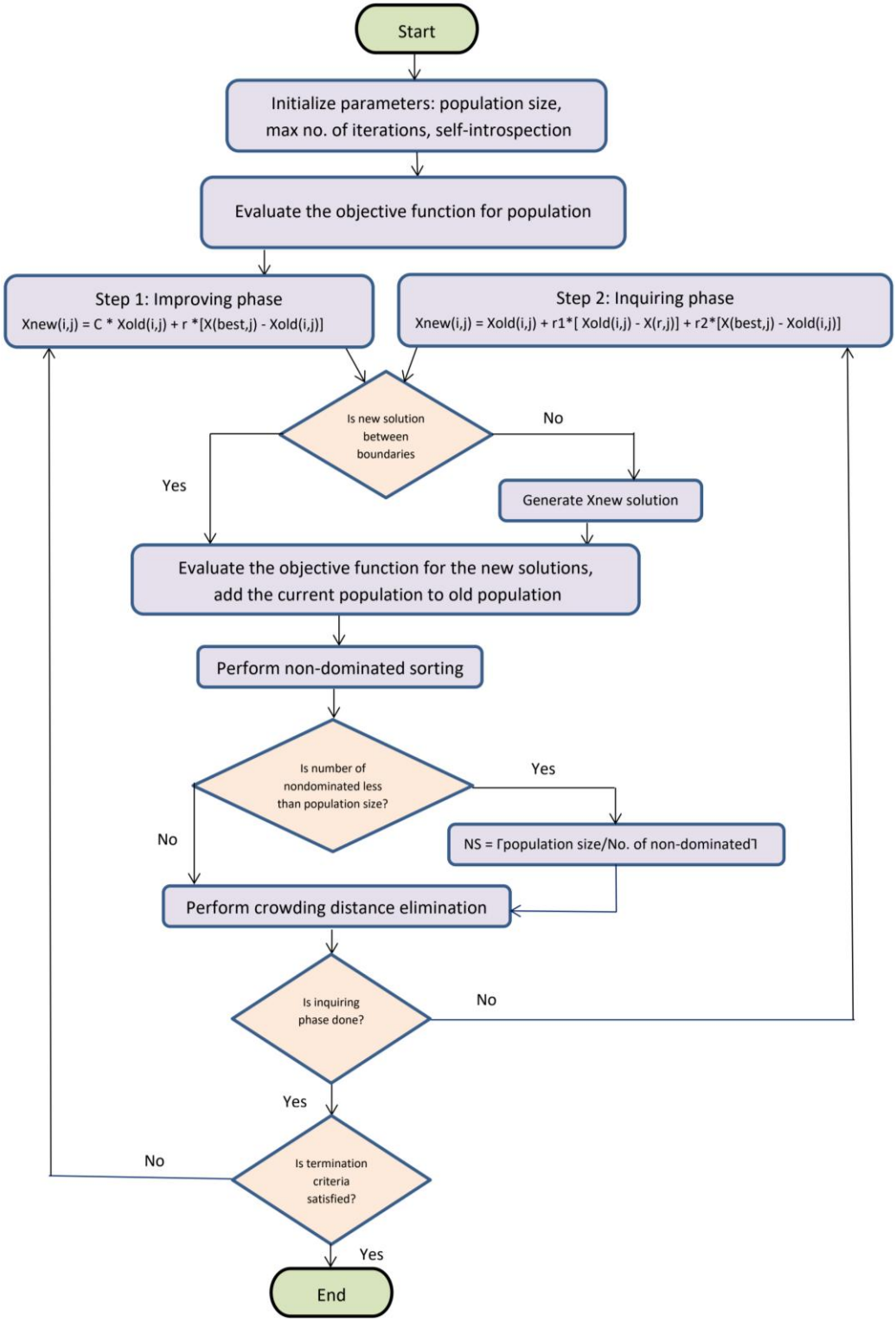

Figure 6. Flowchart of MOSGO. The improving and inquiring phases are executed sequentially within each iteration, although they are shown side-by-side for presentation purposes.

The convergence properties of the proposed MOSGO algorithm can be studied in the stochastic multi-objective optimization framework. The algorithm has three crucial mechanisms: iteration-dependent objective emphasis strategy (through Dynamic Fitness Function, DFF), elitism (through non-dominated sorting), and diversity preservation (via crowding distance).

***Proposition* 1 (*Adaptive Selection via DFF*)**
The Dynamic Fitness Function (DFF) induces an iteration-dependent objective emphasis strategy that balances exploration and exploitation.

***Justification*:**
The Dynamic Fitness Function (DFF) adaptively controls the contribution of objective values in the search process. Specifically, a cyclic objective-guided selection strategy is applied, where a different objective is emphasized at different iterations to define the guiding solution (X_{best}). For minimization problems, the solution minimizing the selected objective for the current iteration is chosen as (X_{best}). However, its complete objective vector is preserved during population evaluation. For instance, the algorithm in the first iteration will concentrate on the first objective to guide the search towards solutions with improved (f_1) values. In later iterations, the focus is shifted periodically to the remaining objectives.

This mechanism fosters the generation of offspring solutions that progressively improve different objectives across generations. This results in exploring multiple trade-off regions of the Pareto front, rather than focusing prematurely on a single objective direction. Theoretically, the DFF introduces a cyclic adaptive selection pressure, where the search process is driven by different objectives at different stages of evolution. The adaptive behavior improves the exploration ability, the coverage of the Pareto front, and reduces the probability of early convergence to local Pareto-optimal regions.

***Proposition* 2 (*Monotonic Improvement of Non-Dominated Set*)**
Let (NDS_t) denote the set of non-dominated solutions at iteration (t). Under the elitist selection mechanism, the sequence ($\{NDS_t\}$) satisfies:
$[NDS_t \preccurlyeq NDS_{t+1}, \forall t \geq 0]$
where, ($\preccurlyeq$) denotes Pareto-set dominance.

***Justification*:**
The candidate solutions generated at each iteration by the improving and inquiring phases are evaluated using the Pareto dominance. The non-dominated sorting procedure makes sure that all non-dominated solutions are kept. Thus, any solution in (NDS_t) is dominated by some solution in (NDS_{t+1}) unless replaced by a dominating solution. Therefore, the quality of the non-dominated set cannot deteriorate over time. Furthermore, the stochastic nature of the search process allows a non-zero probability to generate improved solutions in unvisited regions. Therefore, the algorithm can converge to the Pareto-optimal front in the limit.

***Proposition* 3 (*Diversity Preservation via Crowding Distance*)**
The crowding distance mechanism favors solutions in less crowded regions of the objective space, promoting diversity.

***Justification*:**
Crowding distance estimates the size of the hyper-rectangle that contains each solution. Solutions with larger crowding distance values are farther away from their neighbours. The selection process promotes the elimination of solutions with smaller crowding distances, reducing clustering and keeping a good distribution of the approximation of the Pareto front.

To conclude, the mixture of adaptive selection, elitism, and diversity preservation meets the fundamental requirements for effective convergence and diversity in evolutionary multi-objective optimization. The proposed MOSGO framework guarantees the gradual improvement of the solution quality while preserving diversity. Furthermore, these properties are in agreement with the convergence behavior of the existing multi-objective evolutionary algorithms.

4. Computational Experiments

4.1 Benchmark Test Functions

Zitzler et al. (2000) introduced six well-chosen test functions to compare evolutionary algorithms in solving multi-objective problems. Each test function has a specific feature that makes it difficult for evolutionary algorithms to converge to Pareto front solutions. ZDTs are the most commonly used set to evaluate the performance of evolutionary algorithms. **Table 1** presents the ZDT benchmark function set across many dimensions. The table also provides the formulas for these functions, the domains of the variables, and a description of the shape of the true Pareto front. Note that ZDT5 has often been omitted from analyses in the EA literature (Sharifi et al., 2021) because it is binary-encoded.

Table 1. ZDT benchmark test functions (ZDT5 excluded; see text).

Function	Dimension	Formula	Variable bounds	Pareto front
ZDT1	30	$f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \sqrt{\frac{x_1}{g(x)}}\right)$ $g(x) = 1 + \frac{9\sum_{i=2}^{d} x_i}{d-1}$	$x_i \in [0, 1]$ i = 2,...,d	Convex
ZDT2	30	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - \left(\frac{x_1}{g(x)}\right)^2\right]$ $g(x) = 1 + \frac{9\sum_{i=2}^{d} x_i}{d-1}$	$x_i \in [0, 1]$ i = 2,...,d	Non-convex
ZDT3	30	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\sin(10\pi x_1)\right]$ $g(x) = 1 + \frac{9\sum_{i=2}^{d} x_i}{d-1}$	$x_i \in [0, 1]$ i = 2,...,d	Convex, disconnected
ZDT4	10	$f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \sqrt{\frac{x_1}{g(x)}}\right)$ $g(x) = 1 + 10(d-1) + \sum_{i=2}^{d}[x_i^2 - 10\cos(4\pi x_i)]$	$x_1 \in [0, 1]$ $x_2,...,x_d \in [-5, 5]$ i = 2,...,d	Non-convex
ZDT6	10	$f_1(x) = 1 - \exp(-4x_1)sin^6(6\pi x_1)$ $f_2(x) = g(x)\left[1 - \left(\frac{x_1}{g(x)}\right)^2\right]$ $g(x) = 1 + 9\left(\frac{\sum_{i=2}^{d} x_i}{d-1}\right)^{0.25}$	$x_i \in [0, 1]$ i = 2,...,d	Non-convex, Nonuniformly spaced

4.2 Performance Metrics

There is no scientific evidence to support considering any single metric used in multi-objective optimization as the best metric for evaluating the efficiency of algorithms. For this reason, the researchers tend to compare their algorithms using more than one metric. Good value in a specific metric does not always mean

that the algorithm under consideration has high performance. This happens when the algorithm has a good value in a specific metric at the expense of other metrics; that means, it might have bad values in other metrics. **Table 2** describes different types of indicators, including generational distance (GD), spacing (SP), spread (Δ), and hypervolume (HV), which are used to evaluate the proposed algorithm. Additionally, the table shows the formula for each metric, the concerned measure, and a description of these metrics.

In **Table 2,** NPF refers to the number of non-dominated solutions in the obtained PF, and d refers to the Euclidean distance between the i^{th} member in the obtained solutions and the nearest member in the optimal PF. Also, d_h and d_l represent the Euclidean distances between the extreme first and last solutions of the true PF and the extreme first and last solutions of the obtained PF, respectively. The hypervolume is the ratio of the union hypercubes of the obtained solutions with a reference point (r) and the union of the hypercubes of the optimal Pareto front with the same reference point (r).

Table 2. Metrics description.

Metric	Formula	Measure	Description
Generational Distance (GD)	$\frac{1}{NPF}\left(\sum_{i=1}^{NPF} d_i^2\right)^{\frac{1}{2}}$	Convergence	Lower GD means better convergence
Spacing (SP)	$\sqrt{\frac{1}{NPF-1}\sum_{i=1}^{NPF}\left(d_i-\bar{d}\right)^2}$	Diversity	Lower SP means better diversity
Spread (Δ)	$\frac{d_h+d_l+\sum_{i=1}^{NPF-1}\lvert d_i-\bar{d}\rvert}{d_h+d_l+(NPF-1)\bar{d}}$	Spread	Lower Δ means better spread
Hypervolume (HV)	$\frac{Hypercubes\,(PF_{obtained},r)}{Hypercubes(PF_{optimal},r)}$	Convergence and diversity	Higher HV means better overall performance

4.3 Parameter Setting

Multi-objective artificial hummingbird algorithm (MOAHA), multi-objective particle swarm optimization (MOPSO), multi-objective ant lion optimizer (MOALO), and non-dominated sorting genetic algorithm-II (NSGA-II) are employed to find approximate Pareto front solutions for ZDT set functions and compared with the proposed algorithm (MOSGO) using the four common important measure metrics as listed in **Table 2**. **Table 3** presents the parameters for the four algorithms.

Table 3. Parameters tuning.

Algorithms	Parameters
MOALO	population size: N = 100, archive size:N_r = 100, max generation: iter = 100
MOPSO	population size: N = 100, archive size: N_r = 100, max generation: iter = 100, W = 0.4, individual confidence factor: $C1$ = 2, swarm confidence factor: $C2$ = 2, number of grids per each dimension: N_g = 20, maximum vel in percentage: maxvel = 5, uniform mutation percentage: u_{mut} = 0.5
MOAHA	population size: N = 100, archive size: N_r = 100, max generation: iter = 100
NSGA-II	population size: N = 100, max generation: iter = 100, SBX Crossover: η_c =20, Polynomial mutation: η_m = 20, MutationProb=0.5
MOSGO	population size: N = 100, max generation: iter = 100, self-introspection: C = 0.8

For the proposed algorithm, the population size and the maximum number of generations are standardized to align with those employed by the comparative algorithms, thereby ensuring a fair and consistent evaluation. The self-introspection parameter C is determined empirically through preliminary computational experiments, with C = 0.8 yielding the most favorable performance. For the other algorithms,

the parameter configurations provided in their respective MathWorks implementations are adopted, ensuring a reliable and consistent basis for comparison across all benchmark problems.

4.4 Results and Discussion

In this subsection, the experimental results of the proposed method are presented on a set of benchmark problems and compared with state-of-the-art methods. Moreover, the contributions of each component, the sensitivity and robustness of the self-introspection parameter, and the computational complexity are examined to provide a comprehensive validation of the proposed framework.

4.4.1 Comparative Experimental Results

To evaluate the proposed algorithm in solving multi-objective optimization problems to achieve approximate solutions near the true Pareto front, each ZDT test function is solved ten times, independently, using the five algorithms: MOSGO, MOAHA, MOPSO, MOALO, and NSGA-II. Computational simulation results for GD, SP, Δ and HV metrics on ZDT test functions are reported in **Tables 4** to **7**. A comprehensive comparison of all metrics is presented to assess the suggested algorithm.

Table 4. Computational results for GD on ZDTs.

Function	Observation	Algorithm				
		MOSGO	MOAHA	MOPSO	MOALO	NSGA-II
ZDT1	Mean	**8.2785e-04**	8.2909e-04	0.0032	0.0015	0.0490
	Std	3.7475e-05	4.4168e-05	7.8096e-04	8.2753e-04	0.0221
	Best	7.6362e-04	7.6798e-04	0.0024	6.8776e-04	0.0319
	Worst	8.8161e-04	9.0900e-04	0.0050	0.0034	0.1070
ZDT2	Mean	4.9287e-04	4.9731e-04	0.0024	**2.8098e-04**	0.0007
	Std	2.1704e-05	2.6943e-05	0.0011	7.9111e-06	0.0001
	Best	4.5403e-04	4.5298e-04	7.6368e-04	2.6791e-04	0.0005
	Worst	5.2774e-04	5.4323e-04	0.0039	2.8835e-04	0.0009
ZDT3	Mean	0.0028	**0.0024**	0.0042	0.0028	0.0035
	Std	5.7891e-05	1.0386e-04	7.0221e-04	7.6384e-04	0.0004
	Best	0.0027	0.0023	0.0031	0.0018	0.0028
	Worst	0.0029	0.0025	0.0054	0.0043	0.0041
ZDT4	Mean	**8.1962e-04**	8.4908e-04	0.0037	0.0295	0.0137
	Std	5.6960e-05	1.3291e-05	0.0014	0.0606	0.0035
	Best	7.2749e-04	8.2911e-04	0.0024	0.0017	0.0068
	Worst	8.9262e-04	8.7451e-04	0.0072	0.1957	0.0189
ZDT6	Mean	0.0048	0.0243	0.0403	0.1418	**0.0047**
	Std	0.0035	0.0284	0.0173	0.1120	0.0009
	Best	0.0032	0.0032	0.0190	0.0026	0.0034
	Worst	0.0142	0.0682	0.0710	0.2764	0.0070

In terms of the GD metric, **Table 4** presents the mean, standard deviation, best, and worst values, calculated after applying the considered algorithms to the ZDT functions. The MOSGO has the better mean values of GD in 2 out of 5 ZDT test functions (ZDT1 and ZDT4), and the ZDT2 function has the lowest mean value of GD using MOALO equal to 2.8098e-04. The MOAHA yielded the lowest mean GD on ZDT3, while the NSGA-II yielded the lowest mean GD on ZDT6. The Std of GD values were the lowest using MOSGO on ZDT1 and ZDT3 test functions. The lowest Std values for GD on ZDT2 and ZDT4 were obtained using MOALO and MOAHA, respectively. The NSGA-II provides the lowest Std value for ZDT6. The MOALO gave the best GD values for ZDT1, ZDT2, ZDT3, and ZDT6, but the MOSGO showed the best GD value for ZDT4, equal to 7.2749e-04. Regarding the worst values of GD, MOSGO had the best worst values in the ZDT1 function. MOAHA has the best worst value on ZDT3 and ZDT4 test functions. The best worst value for GD on ZDT2 and ZDT6 is obtained using MOALO and NSGA-II, respectively.

The analysis values of the SP metric, recorded in **Table 5**, show that the MOSGO algorithm yields a better mean in optimizing the ZDT1 and ZDT3 test functions. MOALO, MOAHA, and NSGA-II presented a better mean of SP on ZDT2, ZDT4, and ZDT6, respectively. The better Std values of the SP metric are obtained using the MOSGO for ZDT1, ZDT2, and ZDT3, while MOAHA had the better Std value of the SP on ZDT4 equal to 3.0960e-04, which is slightly smaller than 3.1437e-04, which was obtained using MOSGO. The NSGA-II yielded the lowest Std value of the SP on the ZDT6 test function. The MOSGO had the best SP values on ZDT1, ZDT3, and ZDT6. MOALO had the best value for the SP on ZDT2 and ZDT4. The best worst values of SP are obtained using MOSGO for ZDT1 and ZDT3, while for ZDT2, ZDT4, and ZDT6 are obtained using MOALO, MOAHA, and NSGA-II, respectively.

Table 5. Computational results for SP on ZDTs.

Function	Observation	Algorithm				
		MOSGO	MOAHA	MOPSO	MOALO	NSGA-II
ZDT1	Mean	**0.0029**	0.0032	0.0110	0.0075	0.0492
	Std	2.4029e-04	2.7010e-04	0.0019	0.0045	0.0663
	Best	0.0025	0.0028	0.0073	0.0032	0.0096
	Worst	0.0032	0.0037	0.0139	0.0184	0.2165
ZDT2	Mean	0.0030	0.0032	0.0138	**6.2592e-04**	0.0068
	Std	1.9540e-04	2.7621e-04	0.0054	2.9128e-04	0.0007
	Best	0.0027	0.0026	0.0050	3.1719e-04	0.0056
	Worst	0.0033	0.0036	0.0230	0.0011	0.0078
ZDT3	Mean	**0.0036**	0.0106	0.0196	0.0144	0.0107
	Std	2.5934e-04	0.0030	0.0051	0.0083	0.0097
	Best	0.0032	0.0064	0.0106	0.0071	0.0059
	Worst	0.0040	0.0165	0.0291	0.0310	0.0382
ZDT4	Mean	0.0041	**0.0032**	0.0141	0.0192	0.0159
	Std	3.1437e-04	3.0960e-04	0.0036	0.0565	0.0080
	Best	0.0036	0.0027	0.0102	9.1959e-04	0.0065
	Worst	0.0046	0.0038	0.0217	0.1800	0.0365
ZDT6	Mean	0.0092	0.1571	0.1241	0.0551	**0.0058**
	Std	0.0185	0.2776	0.1200	0.0476	0.0011
	Best	0.0021	0.0025	0.0367	0.0150	0.0046
	Worst	0.0617	0.6802	0.4483	0.1618	0.0082

Table 6 introduced the performance of the Δ metric for all algorithms. Regarding the mean values of Δ, it is clear that the MOSGO was dominant for all ZDT functions except that the MOAHA was outperformed on ZDT4. MOSGO had the better Std values in performing ZDT1, ZDT2, and ZDT3. MOAHA gave the better Std on ZDT4 and NSGA-II on ZDT6. The best values of Δ are between the MOSGO (ZDT1, ZDT2, and ZDT3) and MOAHA (ZDT4 and ZDT6). MOSGO gave the best worst values on ZDT1, ZDT2, and ZDT3 functions, while the best worst Δ values for ZDT4 and ZDT6 were produced by applying MOAHA and NSGA-II, respectively.

Riquelme et al. (2015) concluded that the hypervolume is the most used indicator for evaluating the multi-objective algorithms. The obtained results for the HV metric, displayed in **Table 7**, demonstrated that the MOSGO was superior on ZDT3, ZDT4, and ZDT6 functions regarding the mean values of HV. ZDT1 and ZDT2 gave better mean values using MOALO. MOAHA had better Std values for ZDT1, ZDT2, and ZDT4, while MOSGO had better Std values for ZDT3 and ZDT6. The best values of HV come from applying MOALO on ZDT1, ZDT2, and ZDT6. MOSGO had the best HV value on ZDT3 and shared the outperforming with MOAHA on ZDT4. The best worst value of HV can be noticed on ZDT1 and ZDT4 by applying MOAHA, whereas MOSGO produced the better worst values of HV on ZDT3 and ZDT6 test functions. MOALO produced the best worst value of HV on ZDT2.

Table 6. Computational results for Δ on ZDTs.

Function	Observation	Algorithm				
		MOSGO	MOAHA	MOPSO	MOALO	NSGA-II
ZDT1	Mean	**0.7121**	0.7168	0.8229	1.1034	1.0368
	Std	0.0040	0.0044	0.0143	0.0189	0.0648
	Best	0.7045	0.7094	0.7961	1.0777	0.9353
	Worst	0.7165	0.7225	0.8483	1.1303	1.1180
ZDT2	Mean	**0.7123**	0.7140	0.8213	1.0185	0.7763
	Std	0.0039	0.0045	0.0435	0.0052	0.0104
	Best	0.7042	0.7085	0.7417	1.0113	0.7568
	Worst	0.7190	0.7210	0.8636	1.0264	0.7914
ZDT3	Mean	**0.7698**	0.8758	0.8932	1.1465	0.8999
	Std	0.0034	0.0278	0.0166	0.0376	0.0153
	Best	0.7660	0.8412	0.8647	1.0908	0.8776
	Worst	0.7747	0.9282	0.9197	1.1891	0.9248
ZDT4	Mean	0.7212	**0.7175**	0.8169	1.0398	1.1332
	Std	0.0041	0.0022	0.0203	0.0606	0.0219
	Best	0.7167	0.7141	0.7825	0.9886	1.0781
	Worst	0.7307	0.7209	0.8512	1.2033	1.1531
ZDT6	Mean	**0.7548**	0.9345	1.2349	1.1815	0.7795
	Std	0.0697	0.2794	0.0280	0.0759	0.0144
	Best	0.7183	0.7153	1.1980	1.0705	0.7621
	Worst	0.8978	1.3230	1.2944	1.2927	0.8023

Table 7. Computational results for HV on ZDTs.

Function	Observation	Algorithm				
		MOSGO	MOAHA	MOPSO	MOALO	NSGA-II
ZDT1	Mean	0.9727	0.9724	0.8561	**0.9766**	0.9365
	Std	4.8344e-04	4.1522e-04	0.1268	0.0094	0.0198
	Best	0.9733	0.9730	0.9672	0.9843	0.9481
	Worst	0.9716	0.9718	0.6348	0.9583	0.8842
ZDT2	Mean	0.9249	0.9250	0.8041	**0.9473**	0.9236
	Std	3.7980e-04	2.7969e-04	0.1112	0.0013	0.0008
	Best	0.9256	0.9254	0.9021	0.9487	0.9250
	Worst	0.9243	0.9246	0.6527	0.9456	0.9225
ZDT3	Mean	**0.9664**	0.9565	0.6631	0.9300	0.9299
	Std	6.1433e-04	0.0043	0.0988	0.0198	0.0088
	Best	0.9676	0.9608	0.8417	0.9588	0.9413
	Worst	0.9653	0.9475	0.5090	0.9065	0.9115
ZDT4	Mean	**0.9958**	0.9957	0.7286	0.8950	0.9458
	Std	4.0993e-04	3.8595e-04	0.1401	0.1420	0.0030
	Best	0.9965	0.9966	0.9314	0.9735	0.9512
	Worst	0.9952	0.9953	0.5187	0.5117	0.9411
ZDT6	Mean	**0.9456**	0.9398	0.8952	0.6622	0.9358
	Std	0.0022	0.0086	0.0177	0.2689	0.0044
	Best	0.9468	0.9471	0.9160	0.9570	0.9421
	Worst	0.9395	0.9293	0.8666	0.2977	0.9257

The relative efficiency (RE) criterion (see Dolan and Moré, 2002) is exploited to assess the proposed algorithm based on the results obtained in **Tables 4** to **7**. The ratio $R(i,j)$ is defined as the mean value of the specific metric for the i^{th} test problem computed using the j^{th} solver, and is denoted by $M(i,j)$ divided by the mean value of the same specific metric for the i^{th} test problem obtained using the considered solver. The ratio can be calculated as follows:

$$R(i,j) = \frac{M(i,j)}{M(i,considredsolver)} \tag{5}$$

This ratio serves to normalize the performance of each algorithm relative to a reference, enabling a fair comparison across problems with different scales. The relative efficiency RE is the geometric mean of these ratios for the j^{th} solver over all the test problems and can be computed as follows:

$$RE(j) = \left(\prod_{i=1}^{S} R(i,j)\right)^{1/S} \quad (6)$$

where, S refers to the number of the test problems. The geometric mean is adopted to aggregate performance ratios due to its robustness and suitability for normalized multiplicative comparisons.

Table 8 shows the relative efficiency values of the algorithms for the indicators on the ZDT test set. It is clear that MOSGO has the least relative efficiency regarding GD, SP, and Δ values, and has the greatest relative efficiency regarding HV values. MOSGO was placed in the 1st rank, and MOAHA was placed in the 2nd rank, as its results were too close to MOSGO's results, followed by MOPSO, MOALO, and NSGA-II. The comparative results of MOPSO, MOALO, and NSGA-II indicate that each one of these algorithms excels in two out of four performance indicators, demonstrating complementary strengths and no single method dominating all metrics. Note that the RE value of MOALO for the SP metric is considered an outlier because it came from the mean value of SP on ZDT2 using MOALO.

Table 8. Relative efficiency results.

Metrics	MOSGO	MOAHA	MOPSO	MOALO	NSGA-II
Generational Distance (GD)	1.0000	1.3486	4.0354	4.0564	7.4797
Spacing (SP)	1.0000	2.1523	5.3558	**0.0061**	3.0850
Spread (Δ)	1.0000	1.0717	1.2234	1.4942	1.2466
Hypervolume (HV)	1.0000	0.9967	0.8169	0.6228	0.9724

Overall, it is found from **Tables 4** to **8** that the proposed MOSGO algorithm outperformed the other four studied competitive algorithms. As can be seen in **Figures 7** to **11**, the obtained Pareto front using MOSGO was superior and more reliable in solving ZDT set functions. These figures illustrate the comparison between the true Pareto front and the obtained Pareto fronts using the five algorithms to solve the ZDT test functions. The data used to demonstrate these figures is the data that has the best hypervolume value obtained from 10 independent runs for each ZDT test function. It is clear from the figures that MOSGO and MOAHA satisfy the convergence, diversity, and spread for all ZDT functions. MOPSO exhibits good spread for all ZDT functions, but only reasonable convergence and diversity.

MOALO shows good convergence for all ZDT functions; however, it demonstrates poor diversity and spread. NSGA-II has a good spread solution but not a well-solution distribution (Spacing) for most of the ZDT functions, and worse convergence than other comparative algorithms. As a case study, the differences among the indicator values obtained by applying MOALO to ZDT2 show that these indicators do not reflect the real evaluation of the MOALO algorithm, while respecting the MOALO algorithm and indicators. The GD, SP, and HV values indicate that MOALO is the best algorithm for solving the ZDT2 function, whereas the Δ value suggests that MOALO performs worse than other algorithms under consideration. Regarding the Pareto fronts (non-dominated solutions) obtained from applying MOALO to ZDT1 and ZDT3, they do not exhibit adequate spread along the optimal Pareto front (see **Figures 7** and **9**). For ZDT2 and ZDT4 test functions, MOALO-generated Pareto fronts (non-dominated solutions) lack diversity and are spread along the optimal Pareto front (see **Figures 8** and **10**). Similarly, when applied to the ZDT6 function, MOALO produces a Pareto front that lacks diversity along the optimal Pareto front (see **Figure 11**).

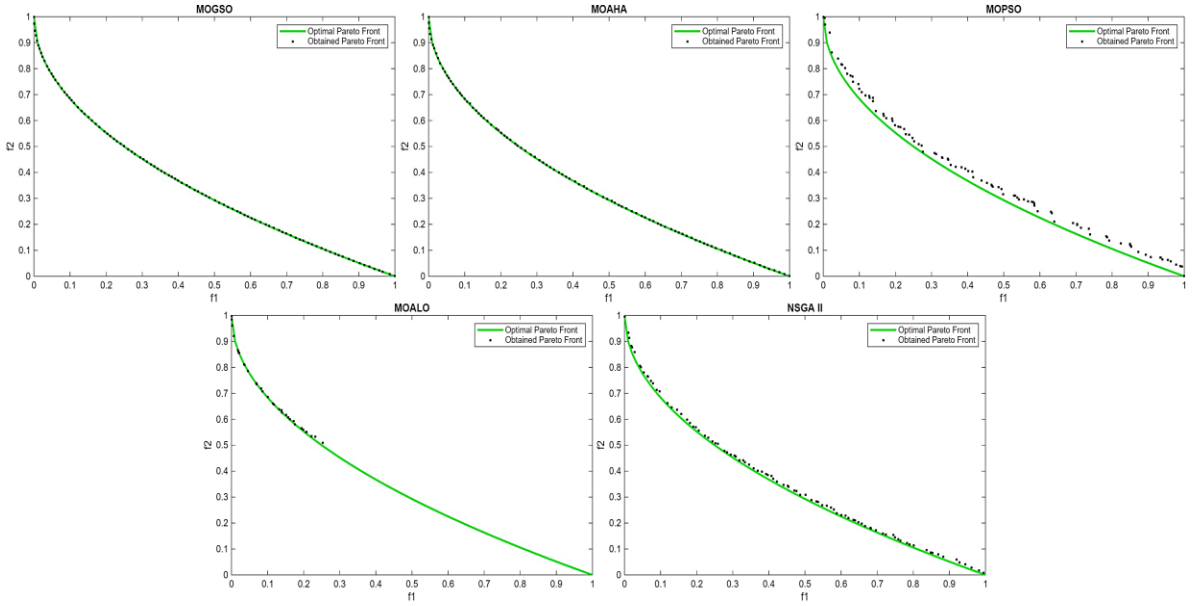

Figure 7. Results obtained using five algorithms on the ZDT1 test function.

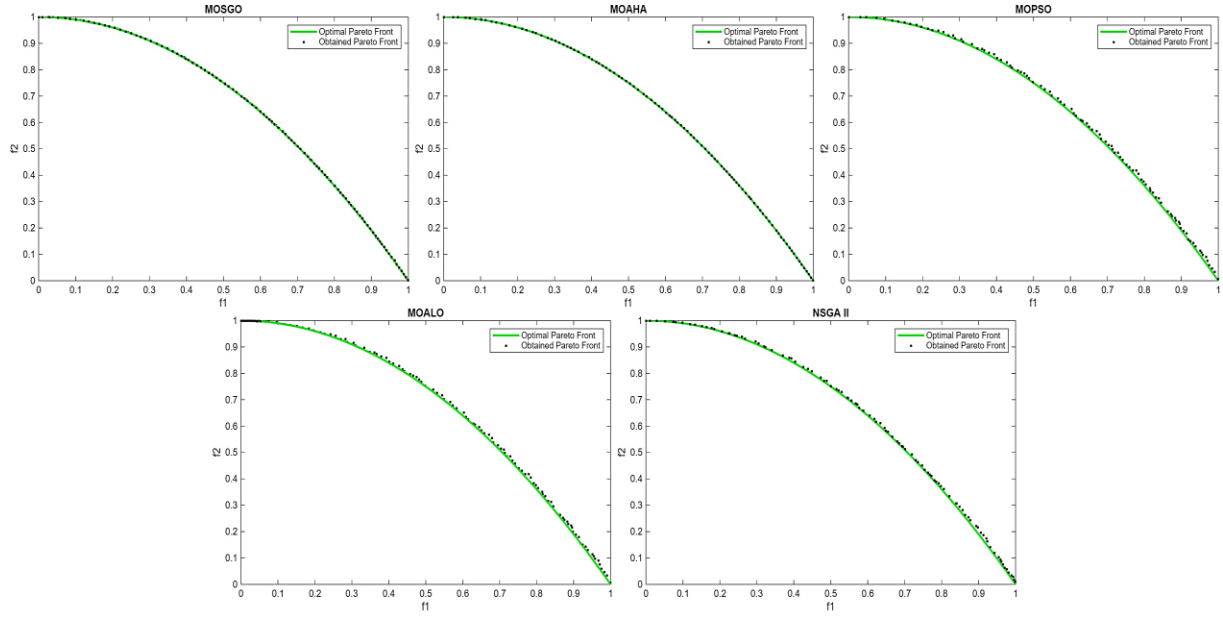

Figure 8. Results obtained using five algorithms on the ZDT2 test function.

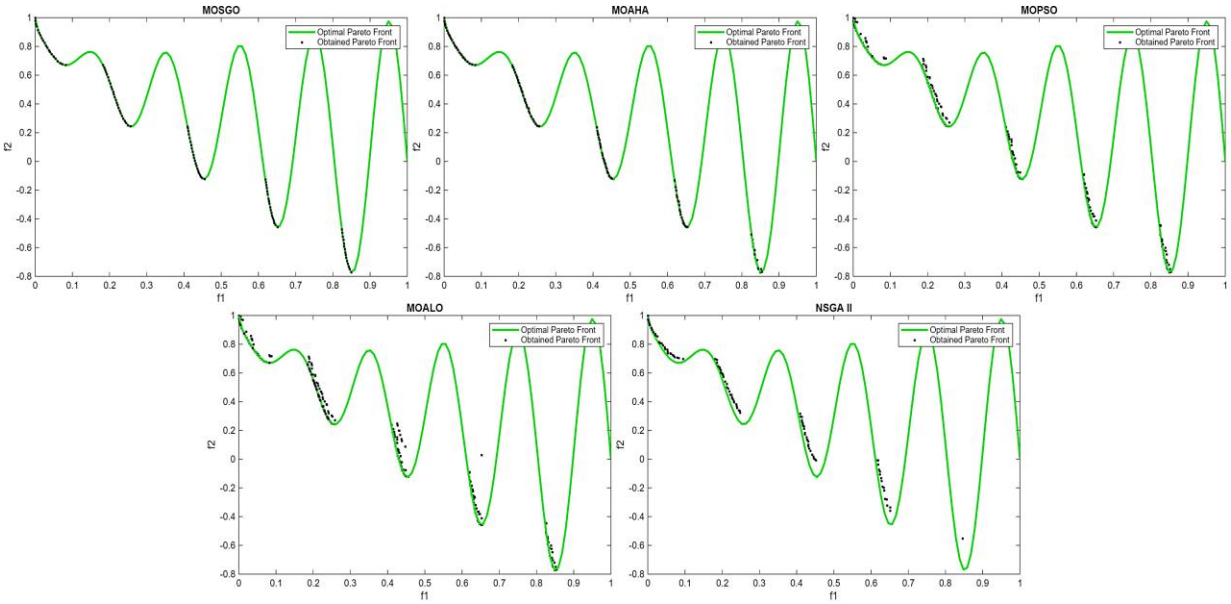

Figure 9. Results obtained using five algorithms on the ZDT3 test function.

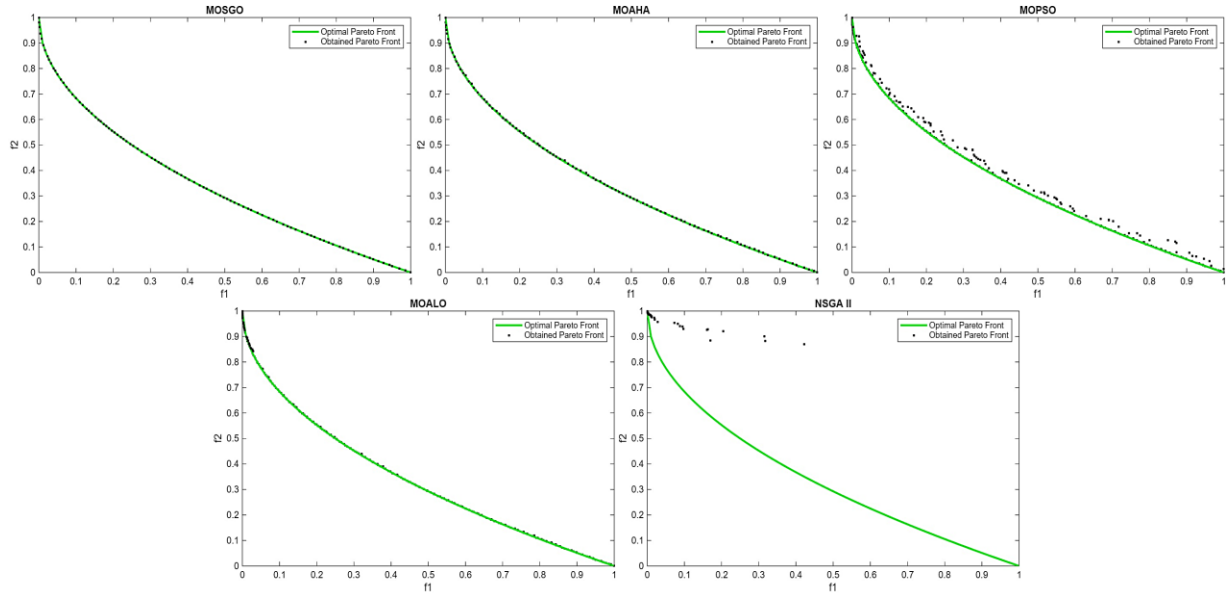

Figure 10. Results obtained using five algorithms on the ZDT4 test function.

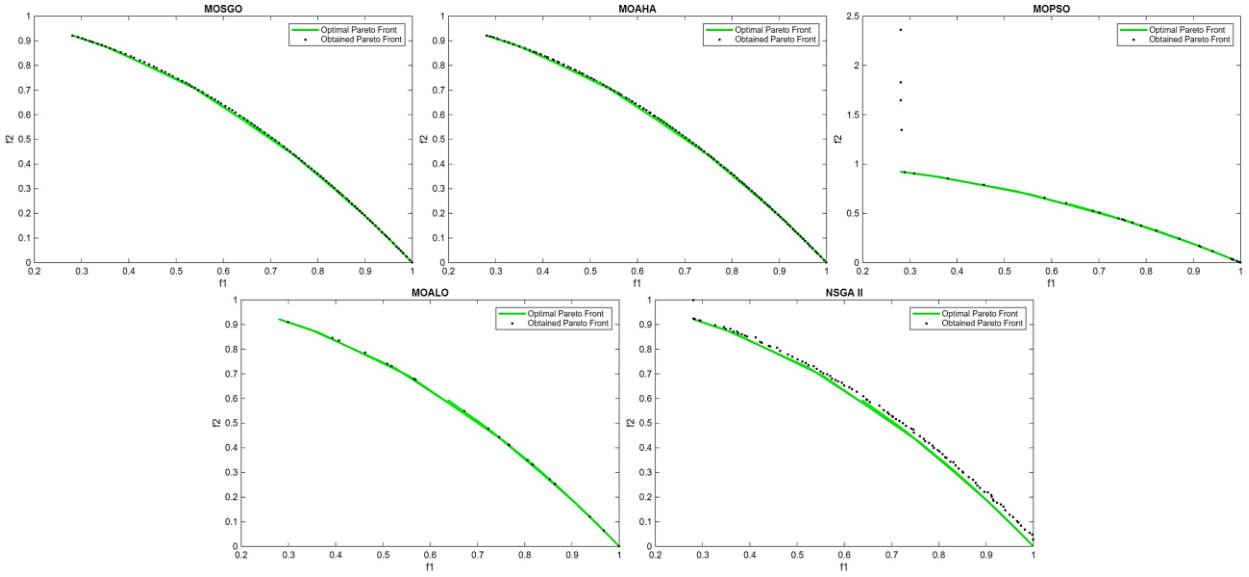

Figure 11. Results obtained using five algorithms on the ZDT6 test function.

Figures 12 to **16** show the boxplots of the generational distance (GD) metric obtained from 10 independent runs of the MOSGO, MOAHA, MOPSO, MOALO, and NSGA-II algorithms to solve the ZDT functions. The statistical analysis of the GD metric on ZDT1 (see **Figure 12**) shows that MOSGO and MOAHA have the lowest median GD values and very narrow boxplots compared to MOPSO, and MOALO. NSGA-II has a slightly wider boxplot than MOPSO, and MOALO. MOAHA ranked first with a median value of 0.00082, followed by MOSGO with a median value of 0.00083. MOALO is located in the third rank with a medium-sized boxplot, followed by MOPSO in the fourth rank. NSGA-II performs worst, exhibiting the highest median GD, a large boxplot, and occupies the fifth rank. **Figure 13** presents the analysis of the GD metric on ZDT2, showing that MOALO is placed in the first rank, as it has the lowest median value and a very narrow boxplot. MOSGO and MOAHA have the same median GD value, which equals 0.00049. Also, they have a small boxplot and are ranked second and third, respectively, depending on their variation (see **Table 4**). The fourth rank is occupied by NSGA-II, with a wider boxplot and a higher median GD metric. Lastly, MOPSO took the fifth rank, characterized by the highest median GD value and the largest boxplot.

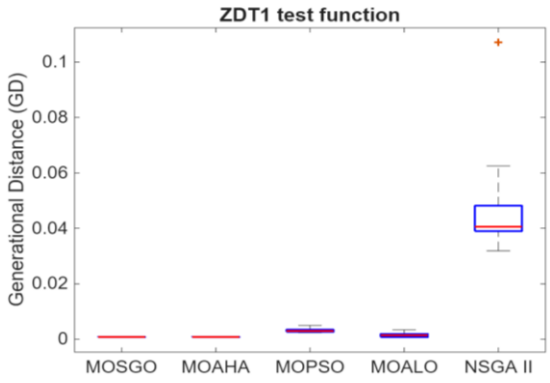

Figure 12. Boxplots of GD on the ZDT1 test function.

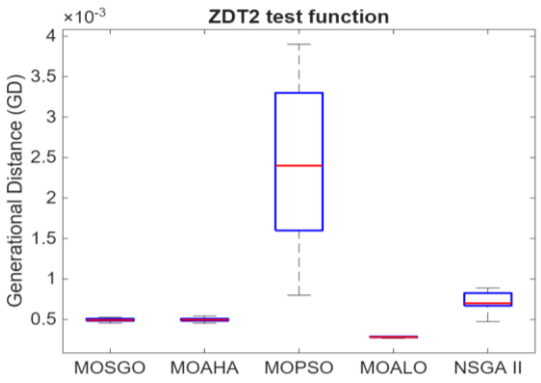

Figure 13. Boxplots of GD on the ZDT2 test function.

Regarding performing the GD metric on ZDT3, **Figure 14** demonstrates that MOSGO and MOAHA have narrower boxplots than MOPSO, MOALO, and NSGA-II. MOAHA has the lowest median GD values, so it took the first rank, followed by MOSGO in the second rank. Then, MOALO, NSGA-II, and MOPSO are located in the third, fourth, and fifth ranks, respectively, based on their median values. **Figure 15** represents the analysis of the GD metric on ZDT4, which shows that MOSGO and MOAHA have the best performance and the smallest boxplots, and are placed in the first and second ranks. Followed by MOPSO, MOALO, and NSGA-II, which are located in the third, fourth, and fifth ranks, respectively, based on their median values. For the ZDT6 benchmark function in **Figure 16**, the boxplot shows that MOSGO and MOAHA have the same lowest median GD values, so they are placed in the first and second ranks, respectively, based on their variation (see **Table 4**). They were followed by NSGA-II and MOPSO in the third and fourth ranks, respectively. MOALO showed the worst performance, as it has the highest median value and the largest boxplot, which led to placing it in the fifth rank.

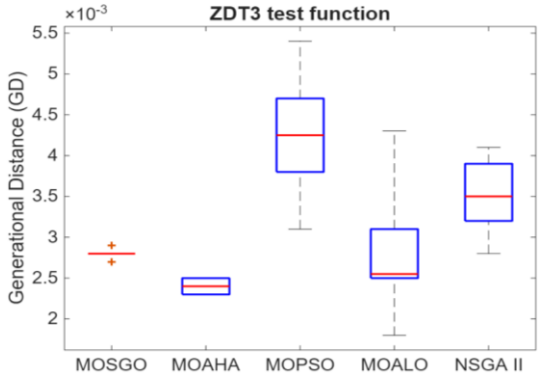

Figure 14. Boxplots of GD on the ZDT3 test function.

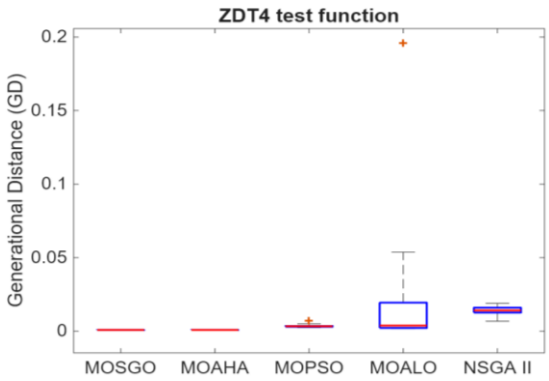

Figure 15. Boxplots of GD on the ZDT4 test function.

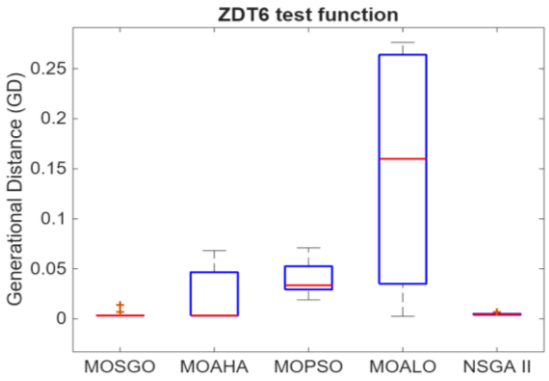

Figure 16. Boxplots of GD on the ZDT6 test function.

To verify the existence of significant differences between the proposed algorithm and the others, a statistical validation is performed using the Wilcoxon signed-rank test. It is the most reasonable non-parametric test for comparing the performance of any two algorithms. For a specific metric, the Wilcoxon signed-rank test applies under the null hypothesis H_0: There is no significant difference between the two algorithms. If the

P-values corresponding to each metric are less than $P < 0.05$, then the H_0 will be rejected. **Tables 9** to **12** below illustrate the corresponding test values and P-values.

Table 9. Wilcoxon signed-rank test comparing MOSGO and MOAHA on ZDT functions according to each metric GD, SP, Δ, and HV.

Function	Metric	Wilcoxon signed-rank			
		MOAHA - MOSGO		Median of MOAHA	Median of MOSGO
		Test Value	P-value		
ZDT1	GD	29	0.9219	0.00082	0.00083
	SP	51	0.01367*	0.00321	**0.00286**
	Δ	46	0.06445	0.71799	0.71377
	HV	13	0.1602	0.97239	0.97273
ZDT2	GD	23	0.6953	0.00049	0.00049
	SP	46	0.06445	0.00327	0.00300
	Δ	28	0.5703	0.71215	0.71146
	HV	28	1	0.92500	0.92480
ZDT3	GD	0	0.00195*	**0.00243**	0.00280
	SP	55	0.00195*	0.01038	**0.00364**
	Δ	55	0.00195*	0.87002	**0.76860**
	HV	0	0.00195*	0.95768	**0.96638**
ZDT4	GD	43	0.1309	0.00085	0.00081
	SP	19	0.4316	0.00317	0.00399
	Δ	8	0.04883*	**0.71718**	0.72018
	HV	37	0.375	0.99561	0.99562
ZDT6	GD	35	0.4922	0.00333	0.00333
	SP	39	0.2754	0.00287	0.00329
	Δ	41	0.1934	0.72661	0.72277
	HV	17	0.3223	0.94604	0.94637

** Since the calculated P-values are less than 0.05, the corresponding null hypothesis will be rejected.*

Table 10. Wilcoxon signed-rank test comparing MOSGO and MOPSO on ZDT functions according to each metric GD, SP, Δ, and HV.

Function	Metric	Wilcoxon signed-rank			
		MOPSO - MOSGO		Median of MOPSO	Median of MOSGO
		Test Value	P-value		
ZDT1	GD	55	0.00195*	0.00301	**0.00083**
	SP	55	0.00195*	0.01134	**0.00286**
	Δ	55	0.00195*	0.82317	**0.71377**
	HV	0	0.00195*	0.91323	**0.97273**
ZDT2	GD	55	0.00195*	0.00242	**0.00049**
	SP	55	0.00195*	0.01309	**0.00300**
	Δ	55	0.00195*	0.83736	**0.71146**
	HV	0	0.00195*	0.86858	**0.92480**
ZDT3	GD	55	0.00195*	0.00422	**0.00280**
	SP	55	0.00195*	0.02010	**0.00364**
	Δ	55	0.00195*	0.89416	**0.76860**
	HV	0	0.00195*	0.65193	**0.96638**
ZDT4	GD	55	0.00195*	0.00335	**0.00081**
	SP	55	0.00195*	0.01298	**0.00399**
	Δ	36	0.43160	0.81927	0.72018
	HV	0	0.00195*	0.74784	**0.99562**
ZDT6	GD	55	0.00195*	0.03361	**0.00333**
	SP	55	0.00195*	0.08881	**0.00329**
	Δ	55	0.00195*	1.23389	**0.72277**
	HV	0	0.00195*	0.90120	**0.94637**

** Since the calculated P-values are less than 0.05, the corresponding null hypothesis will be rejected.*

Table 11. Wilcoxon signed-rank test comparing MOSGO and MOALO on ZDT functions according to each metric GD, SP, Δ, and HV.

Function	Metric	Wilcoxon signed-rank			
		MOALO - MOSGO		Median of MOALO	Median of MOSGO
		Test Value	P-value		
ZDT1	GD	50	0.01953*	0.00143	**0.00083**
	SP	55	0.00195*	0.00680	**0.00286**
	Δ	55	0.00195*	1.09785	**0.71377**
	HV	40	0.23240	0.98044	0.97273
ZDT2	GD	0	0.00195*	**0.00028**	0.00049
	SP	0	0.00195*	**0.00055**	0.00300
	Δ	55	0.00195*	1.01870	**0.71146**
	HV	55	0.00195*	**0.94704**	0.92480
ZDT3	GD	24	0.76950	0.00257	0.00280
	SP	55	0.00195*	0.01100	**0.00364**
	Δ	55	0.00195*	1.14378	**0.76860**
	HV	0	0.00195*	0.92534	**0.96638**
ZDT4	GD	55	0.00195*	0.00384	**0.00081**
	SP	13	0.16020	0.00143	0.00399
	Δ	53	0.00586*	1.02565	**0.72018**
	HV	0	0.00195*	0.95312	**0.99562**
ZDT6	GD	53	0.00586*	0.15993	**0.00333**
	SP	49	0.02734*	0.03172	**0.00329**
	Δ	55	0.00195*	1.18288	**0.72277**
	HV	4	0.01367*	0.64388	**0.94637**

** Since the calculated P-values are less than 0.05, the corresponding null hypothesis will be rejected.*

Table 12. Wilcoxon signed-rank test comparing MOSGO and NSGA-II on ZDT functions according to each metric GD, SP, Δ, and HV.

Function	Metric	Wilcoxon signed-rank			
		NSGA-II - MOSGO		Median of NSGA-II	Median of MOSGO
		Test Value	P-value		
ZDT1	GD	55	0.00195*	0.04060	**0.00083**
	SP	55	0.00195*	0.02045	**0.00286**
	Δ	55	0.00195*	1.03320	**0.71377**
	HV	0	0.00195*	0.94410	**0.97273**
ZDT2	GD	55	0.00195*	0.00070	**0.00049**
	SP	55	0.00195*	0.00665	**0.00300**
	Δ	55	0.00195*	0.77655	**0.71146**
	HV	0	0.00195*	0.92350	**0.92480**
ZDT3	GD	55	0.00195*	0.00350	**0.00280**
	SP	55	0.00195*	0.00760	**0.00364**
	Δ	55	0.00195*	0.89980	**0.76860**
	HV	0	0.00195*	0.93120	**0.96638**
ZDT4	GD	55	0.00195*	0.01415	**0.00081**
	SP	55	0.00195*	0.01455	**0.00399**
	Δ	55	0.00195*	1.13495	**0.72018**
	HV	0	0.00195*	0.94525	**0.99562**
ZDT6	GD	37	0.37500	0.00455	0.00333
	SP	45	0.08398	0.00570	0.00329
	Δ	36	0.43160	0.77735	0.72277
	HV	0	0.00195*	0.93635	**0.94637**

** Since the calculated P-values are less than 0.05, the corresponding null hypothesis will be rejected.*

Table 9 indicates that MOSGO and MOAHA show comparable performance on most ZDT test functions. MOSGO demonstrates statistically significant superiority in several cases, where the P-value is less than 0.05. MOSGO outperforms MOAHA on ZDT1 in the SP metric and on ZDT3 in the SP, Δ, and HV metrics.

These results suggest that MOSGO provides better solution diversity and convergence behavior on multi-objective problems. In **Table 10**, all the P-values are less than 0.05 except for the Δ metric of the ZDT4 function, which means the null hypothesis will be rejected and there are statistically significant differences between the MOSGO and MOPSO algorithms. The metrics values demonstrate that MOSGO significantly outperforms MOPSO across the corresponding ZDT functions. These findings indicate that MOSGO is a more robust and effective multi-objective optimization algorithm compared to MOPSO. Since most of the P-values in **Table 11** are less than 0.05, this means the differences between the MOSGO and MOALO are statistically significant. The metric values indicate that MOSGO outperforms MOALO in the corresponding cases. **Table 12** indicates that MOSGO significantly outperforms NSGA-II on most ZDT test functions. Statistically significant differences ($P < 0.05$) are found for GD, SP, Δ, and HV on ZDT1–ZDT4. In the case of ZDT6, the two algorithms exhibit similar performance across GD, SP, and Δ, while MOSGO achieves a significantly higher hypervolume. These results highlight the superior convergence and solution diversity of MOSGO in most test cases.

4.4.2 Ablation Analysis

To evaluate the individual contribution of the proposed components within the MOSGO framework, an ablation study is conducted. The objective is to determine the quantitative effectiveness of the dynamic fitness function (DFF), the non-dominated sorting (NDS) mechanism, and the crowding distance elimination (CDE) strategy on the overall performance of the algorithm. Four algorithmic variants are considered:

- **MOSGO (Full Model):** The complete proposed algorithm incorporating DFF, NDS, and CDE.
- **MOSGO–DFF:** A reduced version without the DFF, where the solution located in the first rank with the highest crowding distance elimination is used to select the best solution.
- **MOSGO–NDS:** A variant without NDS procedure, where selection is based on the values of the DFF and the CDE.
- **MOSGO–CDE:** A variant implemented the DFF without the CDE mechanism, allowing all non-dominated solutions to survive without diversity control.

These configurations enable isolation of the effect of each component on convergence and diversity.

The ablation experiments are conducted on the ZDT test functions benchmark, covering both convex and non-convex Pareto fronts. Each algorithm variant is executed independently for 10 runs to account for stochastic variability. Performance is evaluated using standard multi-objective metrics, including Generational Distance (GD), Spacing (SP), spread (Δ), and Hypervolume (HV). All empirical tests are executed under identical parameter settings, including population size, number of iterations, and self-introspection ($C = 0.8$), to ensure a fair comparison. The results of the ablation study are summarized in **Table 13**. The full MOSGO algorithm consistently achieves superior performance across all evaluation metrics.

With the Removal of the Dynamic Fitness Function (MOSGO–DFF), a noticeable decrease in convergence is observed, accompanied by higher GD values and a reduced hypervolume. This demonstrates that DFF plays a significant role in directing the search into the promising Pareto-optimal regions by selecting a suitable (X_{best}) at each iteration. The algorithm suffers the most performance degradation without non-dominated sorting (MOSGO–NDS). The resulting solution sets exhibit poor convergence and loss of the Pareto dominance structure, which highlights the importance of elitist selection in multi-objective optimization. The MOSGO–NDS variant was found to obtain slightly better spread (Δ) values in some cases. This behavior is explained by the removal of non-dominated sorting, which results in less selection pressure towards the Pareto-optimal front and a more uniform distribution of the solutions over the objective

space. Therefore, the distances between neighbouring solutions are more regular and hence lead to better (Δ) values. The diversity of the MOSGO-CDE variant is lower, with much worse spacing values and less uniform approximations of the Pareto front. This confirms that the CDE mechanism is needed in order to sustain the solution spread and to prevent premature clustering.

Table 13. Ablation analysis.

Instance	GD	SP	Δ	HV
ZDT1	8.2785e-04	0.0029	0.7121	0.9727
ZDT1-DFF	9.2153e-04	0.0034	0.7105	0.9718
ZDT1-NDS	0.2078	0.0966	0.6303	0.5479
ZDT1-CDE	8.2766e-04	0.0030	0.7120	0.9728
ZDT2	4.9287e-04	0.0030	0.7123	0.9249
ZDT2-DFF	0.0012	0.0040	0.7178	0.9245
ZDT2-NDS	0.3463	0.0980	0.7150	0.2647
ZDT2-CDE	4.9543e-04	0.0031	0.6138	0.9247
ZDT3	0.0028	0.0036	0.7698	0.9664
ZDT3-DFF	0.0076	0.0116	0.8243	0.9568
ZDT3-NDS	0.2747	0.0839	0.6967	0.3916
ZDT3-CDE	0.0028	0.0039	0.7739	0.9655
ZDT4	8.1962e-04	0.0041	0.7212	0.9958
ZDT4-DFF	0.2957	0.0255	1.0790	0.3358
ZDT4-NDS	13.6437	0.8665	0.6189	0.4836
ZDT4-CDE	6.4839e-04	0.0030	0.7754	0.9873
ZDT6	0.0048	0.0092	0.7548	0.9456
ZDT6-DFF	0.0242	0.0044	0.7929	0.8872
ZDT6-NDS	0.5181	0.0857	0.6327	0.3788
ZDT6-CDE	0.0330	0.0866	1.0126	0.9335

Overall, the ablation results demonstrate that each component contributes meaningfully to the performance of MOSGO, with the DFF providing the most significant improvement in convergence and diversity, thereby affecting the hypervolume behavior, while the CDE mechanism ensures diversity preservation. It is clearly seen that the NDS strategy affects the results of all indicators. The synergistic interaction among the components of MOSGO has an effect on the results obtained, in addition to the standard multi-objective mechanisms for MOSGO, which include the improving and inquiring phases. The DFF works with an adaptive search behavior that satisfies the balance between exploration and exploitation, while the NDS ensures that all solution sets converge to the optimal Pareto front, and the CDE guarantees the diversity of the solutions. These findings provide empirical support for the design choices of the proposed algorithm and reinforce its robustness across different problem landscapes.

4.4.3 Parameter Sensitivity and Robustness Analysis

Sensitivity analysis was performed to evaluate the robustness of the proposed MOSGO algorithm in relation to the self-introspection parameter by varying (C) values, taking values of [0.1, 0.3, 0.5, 0.7, 0.8, and 0.9]. The performance was evaluated over ZDT benchmark problems using standard multi-objective metrics: Generational Distance (GD), Spacing (SP), spread (Δ), and Hypervolume (HV). The results summarized in **Table 14** show that the performance of MOSGO is relatively stable over a large range of (C) values, indicating the robustness of the algorithm to parameter variations. More specifically, moderate values of (C) (roughly in the range of [0.7, 0.8]) consistently provide better convergence and diversity performance as lower GD, SP, and Δ values and higher HV are observed.

From these observations, the value (C = 0.8) is chosen as a good trade-off between convergence and diversity. In particular, the small variation in performance across different combinations of parameters shows that the proposed MOSGO algorithm is not too sensitive to the choice of the (C) parameters, which

increases its practical usability. For very small values of (C), the algorithm demonstrates poor convergence behavior because of its insufficient exploitation capability. In contrast, larger values of (C) have a small negative impact on diversity, suggesting stronger selection pressure on some parts of the Pareto front.

Table 14. Self-introspection parameter sensitivity.

Function	Metric	Self-introspection parameter (C)					
		0.1	0.3	0.5	0.7	0.8	0.9
ZDT1	GD	8.3516e-04	8.0950e-04	7.9469e-04	8.0127e-04	8.2785e-04	0.0011
	SP	0.0040	0.0040	0.0039	0.0042	0.0029	0.0033
	Δ	0.7175	0.7158	0.7149	0.7176	0.7121	0.7150
	HV	0.9724	0.9721	0.9524	0.9725	0.9727	0.9704
ZDT2	GD	0.0000	1.9931e-04	4.6675e-04	5.0278e-04	4.9287e-04	0.0014
	SP	0.0000	0.0015	0.0040	0.0032	0.0030	0.0040
	Δ	1.0000	0.8867	0.7138	0.7148	0.7123	0.7232
	HV	0.9524	0.9413	0.9252	0.9247	0.9249	0.9208
ZDT3	GD	0.0028	0.0028	0.0028	0.0028	0.0028	0.0029
	SP	0.0058	0.0050	0.0045	0.0036	0.0036	0.0054
	Δ	0.7994	0.7832	0.7685	0.7656	0.7698	0.7859
	HV	0.9640	0.9648	0.9657	0.9662	0.9664	0.9619
ZDT4	GD	8.0136e-04	8.3088e-04	8.2652e-04	7.8837e-04	8.1962e-04	8.5743e-04
	SP	0.0042	0.0041	0.0041	0.0033	0.0041	0.0044
	Δ	0.7240	0.7197	0.7199	0.7196	0.7212	0.7251
	HV	0.9955	0.9955	0.9957	0.9962	0.9958	0.9948
ZDT6	GD	0.0142	0.0032	0.0033	0.0032	0.0048	0.0946
	SP	0.0694	0.0032	0.0033	0.0032	0.0092	0.0097
	Δ	0.8326	0.7162	0.7209	0.7214	0.7548	1.0654
	HV	0.9425	0.9468	0.9467	0.9468	0.9456	0.7866

4.4.4 Computational Complexity Analysis

The overall computational complexity of the proposed algorithm remains $O(M.N^2)$, where (M) is the number of objectives and (N) is the population size. As the additional costs introduced by the dynamic fitness function (DFF) and the crowding distance elimination (CDE) do not alter the dominant asymptotic behavior. Specifically, the DFF requires only a linear search with complexity $O(N)$, while the upper bound of the CDE procedure incurs a cost of $O(M.N^2\ log\ N)$. Both components are asymptotically dominated by the $O(M.N^2)$ complexity of the non-dominated sorting (NDS) process. Consequently, the integration of these mechanisms does not increase the overall computational complexity of the algorithm. Noting that the computational complexity of the standard Crowding Distance (CD), which is implemented in NSGA-II, is $O(M.N\ log\ N)$ compared with the computational complexity of the Crowding Distance elimination (CDE). However, the runtime of the proposed algorithm is still less than the runtime of MOAHA and MOPSO. Although MOALO has a lower runtime, it is considered an unreliable algorithm as it gives lower performance results than the other algorithms. See **Table 15**.

Table 15. Computational cost comparison (runtime analysis in seconds).

Function	MOSGO	MOAHA	MOPSO	MOALO	NSGA-II
ZDT1	17.3557	26.1601	18.8163	4.8435	15.0795
ZDT2	16.4490	31.2191	19.2121	4.3504	15.0421
ZDT3	12.0390	26.4615	17.0098	4.0747	14.4514
ZDT4	15.0960	25.9554	18.6203	4.0680	13.5374
ZDT6	14.3432	31.3308	19.3339	4.0163	14.2130

5. Conclusions, Limitations, and Future Work

This study proposed the Multi-Objective Social Group Optimization (MOSGO) algorithm to find approximated solutions to the optimal Pareto front for problems with multiple conflicting objectives. A key innovation of this work is the introduction of the Dynamic Fitness Function (DFF), which evaluates a different objective at each iteration. By combining DFF with Non-Dominated Sorting (NDS) and Crowding Distance Elimination (CDE) strategies, the algorithm guarantees improvement in at least one objective per iteration, removes weak candidates, refines convergence, and maintains a well-distributed, diverse Pareto front. The proposed algorithm was tested using ZDT benchmark functions. Computational results demonstrated that MOSGO outperforms four multi-objective algorithms (MOAHA, MOPSO, MOALO, and NSGA-II) in terms of convergence, diversity, and solution distribution, as measured by well-known indicators including Generational Distance (GD), Spacing (SP), Spread (Δ), and Hypervolume (HV).

Despite the promising computational results, several limitations should be acknowledged. First, the current implementation and validation of the MOSGO algorithm are restricted to bi-objective optimization problems, which may limit the generalizability of the results to higher-dimensional objective spaces. Second, the experimental evaluation is conducted exclusively on the ZDT benchmark suite, which, although widely adopted, does not fully capture the diversity and complexity of real-world optimization landscapes. Furthermore, the study focuses on static optimization scenarios, and no experiments were performed on noisy or dynamic objective functions.

To address these limitations and build upon the successful implementation of the DFF, several directions for future research are identified. The proposed MOSGO algorithm will be expanded to effectively handle multi-objective problems featuring more than two objective functions in future implementations. The suggested dynamic fitness function (DFF) mechanism will be applied to and tested within other existing multi-objective optimization algorithms to evaluate its broader effectiveness in solving multi-objective problems. Because current available metrics can yield unreasonable evaluations, there is a critical need to review multi-objective performance indicators. Future studies will focus on analyzing multi-objective optimization metrics to develop or identify a reliable, reasonable standard metric for accurately evaluating algorithm performance.

Conflicts of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

The authors are very grateful to the University of Mosul/College of Computer Science and Mathematics for their facilities, which helped improve the quality of this work.

AI Disclosure

The authors declare that no assistance was taken from generative AI to write this article.

References

Akbari, R., Hedayatzadeh, R., Ziarati, K., & Hassanizadeh, B. (2012). A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation*, *2*, 39-52. https://doi.org/10.1016/j.swevo.2011.08.001

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., & Salomon, L. (2021). Performance indicators in multiobjective optimization. *European Journal of Operational Research*, *292*(2), 397-422. https://doi.org/10.1016/j.ejor.2020.11.016

Coello, C.A.C. (2006). Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine, 1*(1), 28-36. https://doi.org/10.1109/MCI.2006.1597059

Coello, C.A.C., Brambila, S.G., Gamboa, J.F., & Tapia, M.G.C. (2021). Multi-objective evolutionary algorithms: past, present, and future. In: Pardalos, P.M., Rasskazova, V., Vrahatis, M.N. (eds) *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*. IEEE. Springer International Publishing, Cham, pp. 137-162. https://doi.org/10.1007/978-3-030-66515-9_5

Coello, C.A.C., Pulido, G.T., & Lechuga, M.S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256-279. https://doi.org/10.1109/TEVC.2004.826067

Corne, D.W., Knowles, J.D., & Oates, M.J. (2000). The Pareto envelope-based selection algorithm for multiobjective optimization. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.P. (eds) *Parallel Problem Solving from Nature*. Springer, Berlin, Heidelberg, pp. 839-848. https://doi.org/10.1007/3-540-45356-3_82

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182-197. https://doi.org/10.1109/4235.996017

Dolan, E.D., & Moré, J.J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, *91*(2), 201-213. https://doi.org/10.1007/s101070100263

Falcón-Cardona, J.G., & Coello, C.A.C. (2020). Indicator-based multi-objective evolutionary algorithms: a comprehensive survey. *ACM Computing Surveys*, *53*(2), 1-35. https://doi.org/10.1145/3376916

Fonseca, C.M., & Fleming, P.J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, *3*(1), 1-16. https://doi.org/10.1162/evco.1995.3.1.1

Jangir, P., & Jangir, N. (2017). Non-dominated sorting whale optimization algorithm (NSWOA): a multi-objective optimization algorithm for solving engineering design problems. *Global Journal of Researches in Engineering: F Electrical and Electronics Engineering*, *17*(4), 15-42.

Lai, X., Li, C., Zhang, N., & Zhou, J. (2019). A multi-objective artificial sheep algorithm. *Neural Computing and Applications*, *31*(8), 4049-4083. https://doi.org/10.1007/s00521-018-3348-x

Liu, J., & Chen, X. (2019). An improved NSGA-II algorithm based on crowding distance elimination strategy. *International Journal of Computational Intelligence Systems*, *12*(2), 513-518. https://doi.org/10.2991/ijcis.d.190328.001

Marler, R.T., & Arora, J.S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*(6), 369-395. https://doi.org/10.1007/s00158-003-0368-6

Mirjalili, S., Jangir, P., & Saremi, S. (2017). Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. *Applied Intelligence*, *46*(1), 79-95. https://doi.org/10.1007/s10489-016-0825-8

Mirjalili, S., Saremi, S., Mirjalili, S.M., & Coelho, L.D.S. (2016). Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. *Expert Systems with Applications, 47*, 106-119. https://doi.org/10.1016/j.eswa.2015.10.039

Naik, A., Jena, J.J., & Satapathy, S. (2021). Non-dominated sorting social group optimization algorithm for multi-objective optimization. *Journal of Scientific & Industrial Research*, *80*(02), 129-136.

Ning, J., Zhang, C., Sun, P., & Feng, Y. (2018). Comparative study of ant colony algorithms for multi-objective optimization. *Information*, *10*(1), 11. https://doi.org/10.3390/info10010011

Riquelme, N., Von Lücken, C., & Baran, B. (2015). Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference* (pp. 1-11). IEEE. Arequipa, Peru. https://doi.org/10.1109/CLEI.2015.7360024

Satapathy, S., & Naik, A. (2016). Social group optimization (SGO): a new population evolutionary optimization technique. *Complex & Intelligent Systems, 2*(3), 173-203. https://doi.org/10.1007/s40747-016-0022-8

Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (pp. 93-100). Psychology Press. Pittsburgh, Pennsylvania, USA.

Shang, K., Ishibuchi, H., He, L., & Pang, L.M. (2021). A survey on the hypervolume indicator in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, *25*(1), 1-20. https://doi.org/10.1109/TEVC.2020.3013290

Sharifi, M.R., Akbarifard, S., Qaderi, K., & Madadi, M.R. (2021). A new optimization algorithm to solve multi-objective problems. *Scientific Reports*, *11*(1), 20326. https://doi.org/10.1038/s41598-021-99617-x

Sharma, S., & Kumar, V. (2022). A comprehensive review on multi-objective optimization techniques: past, present and future. *Archives of Computational Methods in Engineering*, *29*(7), 5605-5633. https://doi.org/10.1007/s11831-022-09778-9

Xiao, Z., Zhang, M., Liang, Z., Wang, J., Zhu, Y., Li, B., Hu, Y., Wang, J., & Jiang, X. (2024). Improved multi-objective butterfly optimization algorithm and its application in cascade reservoirs optimal operation considering ecological flow. *Water Resources Management*, *38*(12), 4803-4821. https://doi.org/10.1007/s11269-024-03889-7

Yang, X.S., & Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, *40*(6), 1616-1624. https://doi.org/10.1016/j.cor.2011.09.026

Zhang, Q., & Li, H. (2007). MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *11*(6), 712-731. https://doi.org/10.1109/TEVC.2007.892759

Zhao, W., Zhang, Z., Mirjalili, S., Wang, L., Khodadadi, N., & Mirjalili, S.M. (2022). An effective multi-objective artificial hummingbird algorithm with dynamic elimination-based crowding distance for solving engineering design problems. *Computer Methods in Applied Mechanics and Engineering*, *398*, 115223. https://doi.org/10.1016/j.cma.2022.115223

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: methods and applications*. Ithaca: Shaker.

Zitzler, E., & Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: the strength pareto approach. *TIK Report 43*.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation*, *8*(2), 173-195. https://doi.org/10.1162/106365600568202