International Journal Of
Industrial Electronics Control and Optimization
Print ISSN: 3517-2645
Online ISSN: 3568-2645
Homepage: https://ieco.usb.ac.ir/

Addressing Dependent Data in Constrained Optimization Problems: A WOA-based Algorithm

Asieh Ghanbarpour[1] | Soheil Zaremotlagh[2] | Fahimeh Dabaghi-Zarandi[3]

Faculty of Electrical and Computer Engineering, University of Sistan and Baluchestan, Zahedan, Iran.[1]
Faculty of Engineering, University of Sistan and Baluchestan, Zahedan, Iran.[2]
Computer Engineering Department, Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran.[3]
Corresponding author's email: ghanbarpour@ece.usb.ac.ir

Article Info

Article type:
Research Article

Article history:
Received: 31-Dec-2023
Received in revised form: 23-March-2024
Accepted:03-April-2024
Published online: 21-June-2024

Keywords:
Compositional data,
Constraint,
Optimization,
Penalty function,
Swarm optimization.

ABSTRACT

Optimization methods are widely used in various fields to find the best possible solution to a given problem through the minimization or maximization of an objective function while adhering to specific constraints. In this paper, we present a new optimization algorithm, called WOADD, which was designed to handle the challenges of constrained optimization problems that involve dependent data. Unlike traditional algorithms that struggle with data dependencies and valid range constraints, WOADD uses a unique normalization process and a dynamic updating mechanism that accurately considers the interdependencies among features. By calculating a scaling parameter to navigate within feasible regions, WOADD adjusts its search strategy to ensure the preservation of data dependencies and adherence to constraints, ultimately leading to more efficient and precise optimization outcomes. Our extensive experimental analysis, which compared WOADD against other swarm optimization methods using a suite of benchmark functions, demonstrates its superior performance in terms of faster convergence rates, improved solution quality, and enhanced determinism in outcomes.

I. Introduction

Optimization methods are widely used in diverse fields, such as engineering, finance, and data science, with the aim of finding the best possible solution to a problem while meeting certain constraints [1-6]. These methods can be broadly classified into two categories: traditional optimization methods [7, 8] and swarm optimization methods [9, 10]. Traditional optimization methods, such as gradient descent [8] and Newton's method [11], rely on the derivatives of the objective function to approach the optimal solution iteratively. These methods are efficient for simpler problems, but may struggle in complex, high-dimensional or noisy scenarios. On the other hand, swarm optimization methods [1, 5, 9, 10, 12-16] are better suited for such complex problems as they explore the search space more extensively, but they can be computationally expensive and require careful parameter tuning for optimal performance.

According to [17], dependent data are "data where the observations are not independent of each other, and hence, the standard statistical methods that assume independence may not be applicable". Dependent variables are common in real-world scenarios across various fields including geochemistry (e.g. mineral compositions) [18, 19], Biology (e.g. microbiome research [20, 21]), tourism (e.g., distribution, share, allocation) [22, 23], society science (e.g. personal behavior) [24, 25], political science (e.g., voting proportions) [26], marketing (e.g. sale data) [27, 28], and Bioinformatics [29].

Assigning weights to the parameters of a function or model is a common problem defined on dependent data. This problem is known as feature weighting in machine learning, and it aims to identify important features and assign weights to the model parameters accordingly [30]. In linear regression, the weights assigned to the predictor variables determine their relative importance in predicting the outcome variable [31]. In neural networks, the weight assigned to each connection between neurons determines the strength of the connection and can significantly impact the network's performance [32, 33]. Optimizing the weights for parameters can be quite a challenge, particularly when dealing with high-dimensional data. One possible solution to address this issue is to use optimization

algorithms [1, 3, 9, 10, 12]. However, it's worth noting that optimization algorithms are primarily designed for unconstrained data, where variables can change independently. To handle constrained data, such as those with dependencies, optimization algorithms employ constraint-handling techniques such as penalty functions, repairing infeasible solutions, and multi-objective optimization.

In this research paper, we present a new method for handling dependent data in optimization systems. Our study shows that penalty functions often lead to convergence problems and suboptimal solutions. Instead, we propose a novel approach that uses the Whale Optimization Algorithm (WOA) to seamlessly integrate constraints into the optimization process. Our algorithm, called WOADD, normalizes the data and dynamically updates the search strategy to consider the interdependencies among the features. It calculates a scaling parameter to maneuver within feasible regions and ensure the preservation of data dependencies and adherence to constraints. Our method has been found to have superior performance in handling optimization problems characterized by dependent data, and it shows potential for more effective and efficient optimization solutions.

Throughout the remainder of this paper, we review related works in Section II, examine the problem statement in Section III, present the proposed method in Section IV, analyze experiments in Section V, and conclude our findings in Section VI.

II. Literature Review

Evolutionary Algorithms (EAs) often struggle to handle constraints directly during optimization in many problems. Various techniques have been used in EAs to solve constrained optimization problems, including penalty functions, repairing infeasible solutions, and multi-objective optimization. An overview of the literature on applying EAs to constraint optimization algorithms is presented in [34].

A penalty function is a method that converts a Constrained Optimization Problem (COP) into an unconstrained one by adding or subtracting a certain value to or from the objective function based on the number of constraint violations present in a given solution [35, 36]. However, determining the appropriate penalty factor for a problem can be challenging, because selecting an excessively high or low penalty can lead to issues [35]. Another group of techniques involves repair algorithms, which aim to transform infeasible solutions into feasible ones by applying specific procedures or mechanisms [37, 38]. Although repair algorithms are useful when converting infeasible solutions to feasible ones is computationally inexpensive, they may introduce biases and adversely impact the evolutionary process. Furthermore, designing a repair algorithm for each problem is a problem-dependent requirement.

A different way of dealing with constraints involves transforming a COP into a problem with m + 1 objectives, where m is the total number of constraints. This problem can then be solved using a multi-objective evolutionary algorithm [39] [40-43]. However, applying multi-objective optimization is not a simple process and requires careful consideration of various factors. One popular approach is to minimize both the original objective function and the degree of constraint violation simultaneously. The paper [39] used a modified non-dominated sorting genetic algorithm after transforming a COP into multi-objective optimization. On the other hand, [40] employed a Pareto-ranking-based Differential Evolution (DE) algorithm. Li et al. [41] addressed many-objective optimization problems through dynamic constraint handling, and [42] converted a COP into a dynamical bi-objective optimization problem. In [43], a multi-objective optimization problem is decomposed into sub-problems using the weighted sum approach, and dynamic adjustments to the weights enable each sub-problem to converge towards an equivalent objective. Recent advancements include techniques such as feasible-ratio control and dynamic constraint boundaries [44] to enhance EAs with dynamic constrained multi-objectives.

III. Problem Statements

A. Problem Definition

Let $\mathfrak{R}^D$ be the feasible region of the search space, contained within the D-dimensional rectangular space S. Denote $X = [X_1, \dots, X_f, \dots, X_D]$ as the feasible solution vector, where X is an element of $\mathfrak{R}^D$, where $\mathfrak{R}^D \subseteq S$. Each component of the feasible solution vector X, denoted as X_f, must satisfy the upper and lower boundary limits X_{fL} and X_{fU}, respectively, where $X_{fL} \le X_f \le X_{fU}$. The objective function value of solution X is defined as $f(X)$, while the functions $g_j(X)$ and $h_j(X)$ represent the *j*-th equality constraint and inequality constraint, respectively. A COP is defined as follows:

$$\min_{X=[X_1,\dots,X_f,\dots,X_D]\in\mathfrak{R}^D} f(X) \quad s.j.\begin{cases} g_j(X) \le 0, j = 1, \dots, p \\ h_j(X) = 0, j = 1, \dots, q \end{cases} \tag{1}$$

, where p and q represent the numbers of inequality and equality constraints of problem respectively.

In the problem of assigning weights to the parameters of a function, the objective is to assign weights to different parameters of a function such that the sum of the weights equals a constant κ, and each weight falls within the range of zero to one. This form of optimization problem can be defined as follows:

$$\min_{X=[X_1,\dots,X_f,\dots,X_D]\in\mathfrak{R}^D} \sum_{f=0}^{n} X_f \times V_f \quad s.j\begin{cases} X_f \ge 0, \ \ f = 0, \dots, n \\ \sum_{f=0}^{n} X_f = \kappa \end{cases} \tag{2}$$

where X_f shows the weighting parameter for $0 \le f \le n$, and V_f is the value of the element which should be weighted by X_f.

B. Whale Optimization Algorithm (WOA)

WOA is an algorithm that follows the social behavior of humpback whales. It works by initializing a set of random solutions (whales) and updating their positions based on whether they are in the exploration or exploitation phase. In

the beginning, each whale is assigned random values. The search process for finding the optimal solution involves two phases: exploration and exploitation. The exploitation phase models the bubble-net behavior of humpbacks when attacking prey. This behavior is modeled with two types of movement: shrinking encircling and spiral updating position. To encircle prey, the positions of search agents are updated towards the best search agent that is assumed to be the target or near the target. This update of the search agents is represented by the following equations, as in [45]:

$$\vec{D} = \left|\vec{C}.\overrightarrow{W^*}(t) - \vec{W}(t)\right| \quad (3)$$

$$\vec{W}(t+1) = \overrightarrow{W^*}(t) - \vec{A}.\vec{D} \quad (4)$$

where $\vec{W}(t)$ and $\vec{W}(t+1)$ shows the configuration of an search agent in the i-th iteration and (i+1)-iteration respectively, $\overrightarrow{W^*}(t)$ shows the best search agent regognized until iteration t. Vectors $\vec{A}$ and $\vec{C}$ are coefficient vectors which are calculated as follows:

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a} \quad (5)$$

$$\vec{C} = 2.\vec{r} \quad (6)$$

During iterations, the linearly decreasing vector $\vec{a}$ is reduced from 2 to zero, while the k-dimensional vector $\vec{r}$ has random values within the range of [0, 1]. The method for achieving shrinking encircling involves decreasing the $\vec{a}$ in Eq. (5). Since $\vec{r}$ is a random vector within the range of [0, 1], the values of $\vec{A}$ fall within the range of $[-a, a]$. This model demonstrates how the search agents move within hyper-cubes towards the current best solution.

The spiral updating position simulates the helix-shaped movement of humpback whales to exploit prey. This behavior is modeled as follows:

$$\vec{W}(t+1) = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{W^*}(t) \quad (7)$$

, where b is a constant which is set to one, and l is a random value in $[-1,1]$. Vector $\overrightarrow{D'}$ shows the distance between the search agent and the best current solution (the prey) and is defined as follows:

$$\overrightarrow{D'} = \left|\overrightarrow{W^*}(t) - \vec{W}(t)\right| \quad (8)$$

Humpback whales use two hunting strategies shrinking encircling and spiral updating positions together to efficiently catch their prey. To mimic this behavior, a new parameter (p) is introduced that randomly switches between the two strategies by taking on a random value between 0 and 1. During the exploitation phase, the configuration is updated using the following process:

$$\vec{W}(t+1) = \begin{cases} \overrightarrow{W^*}(t) - \vec{A}.\vec{D} & if\ p < 0.5 \\ \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{W^*}(t) & if\ p \geq 0.5 \end{cases} \quad (9)$$

During the exploration phase, the search agents update their positions based on a randomly selected search agent $\overrightarrow{W^{rand}}(t)$ as follows:

$$\vec{D} = \left|C.\overrightarrow{W^{rand}}(t) - \vec{W}(t)\right| \quad (10)$$

$$\vec{W}(t+1) = \overrightarrow{W^{rand}}(t) - \vec{A}.\vec{D} \quad (11)$$

IV. Methodology

A. Normalization

Optimization algorithms often use randomly generated initial populations. However, when dealing with dependent data, the features of an object are interdependent. To maintain the dependency of the features while still generating a random initial population, a two-step process is proposed for individual initialization. Firstly, the population is randomly generated. Then, each individual is normalized to preserve the data's dependency.

Let $X = [X_1, \dots, X_f, \dots, X_D]$ represents a solution vector, where X is an element of $\Re^D$. Furthermore, let $\varkappa = \{X_p, \dots, X_q\}$, where $\varkappa \subseteq X$ shows a set of features of X that are dependant on each other according to the condition $M_c(\varkappa)$. This condition is an equality condition that takes the form of a sum function similar to those which are common in the weighting problems. To normalize Individual X based on $M_c(\varkappa)$, each $X_i \in \varkappa$ is first subtracted by the minimum value in the set $\varkappa$ as follows:

$$X_i' = X_i - X_{min} \quad (12)$$

By setting the value of $M_c(\varkappa)$ as α, Eq.(13) is used to transform the independent variables of $\varkappa$ into dependent variables [46].

$$(X_i)_N = \frac{\alpha}{\sum_{X_i \in \varkappa} X_i'} \times X_i' \quad (13)$$

If there exists equality conditions $M_{c_1}(\varkappa_1), M_{c_2}(\varkappa_2), \dots, M_{c_k}(\varkappa_k)$ for sets $\varkappa_1, \varkappa_2, .., \varkappa_k$, and these sets have no overlap, then each equality condition can be applied independently to the individuals of the population to obtain individuals that are within the valid domain of the search.

B. Updating Positions

In the WOA, the optimized solution is determined based on the leader individual, which is the best individual in all iterations. During iterations, the position of each individual X_i is updated based on three movements: two based on the position of the leader individual and the third based on the position of a randomly selected individual. While features of an individual are updated independently in the WOA, a filtering step is executed after updating positions to eliminate individuals that don't satisfy constraints for dependent data or other constraints. When the features of an individual are dependent, changes made to one feature may impact the other features. In such cases, because it is unlikely to generate a valid solution through random changes to the features, many solutions will be eliminated during the filtering stage. Therefore, it is crucial to consider this dependency when updating the individuals.

The Whale Optimization Algorithm for Dependent Data (WOADD) employs an appropriate policy for updating data that considers the dependence between features to avoid unnecessary processing. The changes are precisely controlled to preserve the equality constraints, and the features collaborate to satisfy these constraints.

The change in the value of a feature is entirely dependent on the type of whale movement in the corresponding iteration. The magnitude of change in feature $X_i \in X$ is calculated using the following formulas depending on whether it's a

circular movement, spiral movement, or exploratory movement:

$$\Delta X_{circular} = A_i \times |C_i \times X_i^* - X_i| \quad (14)$$

$$\Delta X_{spiral} = |X_i^* - X_i| \times e^{bl} \times \cos(2\pi l) \quad (15)$$

$$\Delta X_{expl} = A_i \times \left|C_i \times X_i^{rand} - X_i\right| \quad (16)$$

Where X_i shows the i-th feature of normalized individual X, b is a constant for defining the shape of the logarithmic spiral, and l is a random number in [-1,1]. The values of A_i and C_i show the i-th element of vectors $\vec{A}$ and $\vec{C}$ calculated by (5) and (6), respectively.

The change of feature $X_i \in \varkappa$ may violate the equlity condition $M_c(\varkappa)$. In this case, any change in one feature is completed with a candidate one. Let $\mathcal{L} \subseteq \varkappa$ shows the set of features that haven't been changed in the current iteration. The members of $\mathcal{L}$ are those that could be chosen as candidates. To ensure equal chances of being chosen, each member of $\mathcal{L}$ is assigned a random probability, and then sorted in descending order based on their random number. The member with the highest probability is checked first for candidacy. Let ΔX represent the amount of change of feature $X_i \in \varkappa$, and X_c represent the feature being considered as a candidate. There are three possible situations that may occur:

- If $\Delta X \geq 0$, an overflow has occurred in set $\varkappa$ based on the equality constraint. To resolve this issue, the candidate element should have the ability to reduce this additional load. If $(X_c - \Delta X)$ satisfies the defined constraints on X_c, then there will be a load transfer between X_i and X_c. If not, the next candidate feature is checked.
- If $\Delta X < 0$, an underflow has occurred in set $\varkappa$. Therefore, the candidate element should have the potential to carry additional load. If $(X_c + |\Delta X|)$ is validated according to the defined constraints, X_c is selected as the candidate for X_i. Otherwise the other potential candidates are checked.
- If a candidate cannot be found for X_i, it is returned to its previous value.

In each iteration, the features in an individual are updated at most once, and a flag is reserved for each feature to indicate its status. At the start of each iteration, all features of an individual are marked as unchanged. If a feature is modified or checked for an update and no candidate is found, it is marked as changed. Load transfers are continued until all the involved features are marked as changed.

In order to avoid exceeding the valid range of features, a parameter is introduced to limit the range of change for each feature. This parameter's value is determined based on the constraints of the problem and the specific requirements of the application. For simplicity, let all $X_i \in \varkappa$ satisfy $L \leq X_i \leq U$. The mathematical formulation of circular and exploratory movements are similar in that the position of an individual is updated based on a goal individual X^G which can be either the leader individual or a randomly selected individual. Let X_i^G denote the value of the i-th feature in X^G. The next value of X_i in the circular and exploratory movements is obtained as follows:

$$X_i(t+1) = X_i^G - \Delta X_i \quad (17)$$

, where the inequality conditions $L \leq X_i(t+1) \leq U$ should be met. Since $\Delta X_{expl} \geq 0$ and $\Delta X_{circle} \geq 0$ and X^G is an individual in the valid search space, the unequality $X_i(t+1) \leq U$ is always true. To have more control on the movement of objects, we consider a scaling parameter $\mu > 0$ on the change values, and solve the following unequality:

$$X_i^G - \mu \Delta X_i \leq U \quad (18)$$

This implies that:

$$\mu \leq \frac{X_i^G - L}{A\left|CX_i^G - X_i\right|} \quad (19)$$

On the other hands, when completing the change of X_i with candidate X_c, the following unequalities should be satisfied for X_c:

$$L \leq X_c + \mu \Delta X_i \leq U \quad (20)$$

Since $\Delta X_{expl} \geq 0$ and $\Delta X_{circle} \geq 0$ and X_c is in the valid range, the unequality $L \leq X_c + \mu \Delta X_i$ is always true and the unequality $L \leq X_c + \mu \Delta X_i$ should be examined. Solving this implies that:

$$\mu \leq \frac{U - X_c}{A\left|CX_i^G - X_i\right|} \quad (21)$$

Both the equations (19), (21) should be satisfied, therefore μ is determined as follows:

$$\mu \leq \frac{1}{A\left|CX_i^G - X_i\right|} \min\left(X_i^G - L\,, U - X_c\right) \quad (22)$$

As an example, if any variable falls within the range of [0,1], the maximum value of A and C is 2, and the maximum distance between two features of two whales is 1. Therefore, the range of μ would be as follows:

$$0 < \mu \leq \min\left(\frac{X_i^G}{4}, \frac{1 - X_i^G}{4}\right) \quad (23)$$

In the spiral movement, the sign of $\Delta X_{spriral}$ is relative to the sign of $\cos(2\pi l)$. If the value of feature X_i after spiral movement be equal to $\left(X_i^* + \mu \Delta X_{spiral}\right)$, the following unequalities should be met:

$$L \leq X_i^* + \mu(|X_i^* - X_i| \times e^{bl} \times \cos(2\pi l)) \leq U \quad (24)$$

By solving the inequalities, μ is calculated as follows:

$$\begin{cases} \mu \leq \dfrac{U - X_i^*}{|X_i^* - X_i| \times e^{bl} \times \cos(2\pi l)} & \text{if } \cos(2\pi l) \geq 0 \\ \mu \leq \dfrac{X_i^* - L}{|X_i^* - X_i| \times e^{bl} \times |\cos(2\pi l)|} & \text{if } \cos(2\pi l) < 0 \end{cases} \quad (25)$$

To ensure that the given set of inequalities are satisfied, μ is chosen accordingly.

$$0 < \mu \leq \frac{1}{|X_i^* - X_i| \times e^{bl} \times |\cos(2\pi l)|} \min(U - X_i^*, X_i^* - L) \quad (26)$$

It is noted that the candidate feature $X_c$in a spiral movement is changed to $\left(X_c - \mu \Delta X_{spiral}\right)$ which must fall within the range of L and U. However, this constraint also leads to the same range for μ as given in equation (26). The pseudo-code of the proposed algorithm is shown in Figure 1.

Algorithm 1: WOADD

Initialize the Whale population $W_i, i = 1, \dots, n$
Normalize the initial population
Initialize a, A, C, b , l
while $t < number\ of\ iteration$ **do**
  Calculate the fitness of each Whale
  Select the best Whale as the Leader Whale
  for $each\ Whale\ (X)$ **do**
    Randomly initialize r, p
    Update a , A , C, μ
    for each feature (X_i) in X **do**
      if ($p < 0.5$ and $|A| < 1$) **do**
        Calculate ΔX by (14)
      if ($p < 0.5$ and $|A| \geq 1$) **do**
        Select a random whale (X^{rand})
        Calculate ΔX by the (15)
      if ($p \geq 0.5$) **do**
        Calculate ΔX by the (16)
      X_c =Find a candidate for X_i
      if ($X_c! = null$)
        $X_i = X_i + \mu\Delta X$
        $X_c = X_c - \mu\Delta X$
  $t = t + 1$
return LeaderWhale

Fig. 1. Pseudo-code of the WOADD algorithm

V. Test Results

A. Benchmark functions

The performance of optimization algorithms is usually evaluated using a set of 24 benchmark functions [2, 3, 12, 14, 47, 48], as mentioned in CEC 2005 [49]. However, these functions aren't designed to account for dependent data. To test the proposed algorithm, we used a set of functions in Table 1 that include constraints imposing dependence among features. The focus is on minimizing the objective, and the last column of the table shows the optimal values for each function. As previously mentioned, most optimization algorithms lack explicit policies for handling problem constraints. Instead, they rely on penalty functions to manage them. The middle column of Table 1 shows the updated functions based on constraint penalties.

To evaluate optimization algorithms, two measures are commonly used: average fitness value and standard deviation. Suppose the initial population contains n objects. The average fitness value is obtained by summing the fitness value of each member of the population and dividing by the population size. The standard deviation is computed as the square root of the sum of squared distances of each fitness value from the average fitness value, divided by the population size minus one. The standard deviation represents the diversity among the fitness values of the population members. If the standard deviation is very small, it indicates that all search agents are near each other. In such cases, the algorithm may find it difficult to find the best solution, particularly if there are several local optima available.

To demonstrate the effectiveness of the proposed algorithm (WOADD), we conducted a comparative analysis with three other state-of-the-art optimization algorithms, namely Whale Optimization Algorithm (WOA) [45], Gray Wolf Optimization Algorithm (GWO) [12], and Aquila optimizer [10]. In all experiments, the number of search agents is set to 30, and the number of iterations is set to 300. We use state-of-the-art methods with default parameter values, but a penalty function is added to penalize invalid solutions, as shown in Table 1. The penalty function uses α, β, and γ values of 10, 10, and 20, respectively. The results of this comparison are presented in Table 2 for solutions with five and ten dimensions. For each test case, this table displays the algorithm's optimal $f(x)$ value, along with the average and standard deviation of $f(x)$ obtained by various search agents employed by the algorithm. As heuristic algorithms produce non-deterministic solutions, the presented results are the average of 20 runs.

The results show that the proposed algorithm has successfully obtained the minimum value for all functions f_2, f_3, f_4, f_5 and f_6 with particularly significant improvements observed for more complex functions f_5 and f_6. Notably, the proposed method achieved a result approximately 100% better than GWO and about 50% better than WOA for function f_5. However, the low standard deviation observed in GWO is not ideal for population-based approaches as it implies that all search agents are focused on a single target point.

The functions f_5 and f_6 have an additional parameter (c) to be determined in tests. Table 3 shows the performance of the optimization algorithms on these functions for c values of 0.1, 0.3, and 0.5. For instance, when $c = 0.1$, it means that m terms of the series carry 0.1 of the total weight, while the remaining terms carry 0.9 of the total weight. The results demonstrate that the proposed algorithm was able to achieve the best minimum value for the functions in all cases.

B. Analysis of Convergence curve

In population-based evolutionary algorithms, the search agents tend to extensively explore promising regions of the design space and gradually converge toward the best solution. During the initial stages of the optimization process, the search agents undergo abrupt changes, and then gradually converge towards the optimal solution. According to [5], this behavior guarantees that a population-based algorithm eventually converges to a point in the search space. The convergence curves of WOA, GWO, and WOADD for the benchmark functions are compared in Figure 2. Aquila is not included in the figure due to its large values in some cases. The results indicate that the WOADD algorithm is capable of quickly identifying promising areas within the search space during the initial stages of iteration and converges more rapidly towards the optimal solution.

Figure 3 illustrates the diversity of outcomes generated by different algorithms in 20 runs for functions f_5 and f_6. As shown in the figure, WOA and GWO display a high degree of diversity across different runs. In contrast, the results generated by WOADD are very similar to each other, and a consistent outcome is obtained in almost all runs. This makes WOADD more suitable for applications that require more deterministic results.

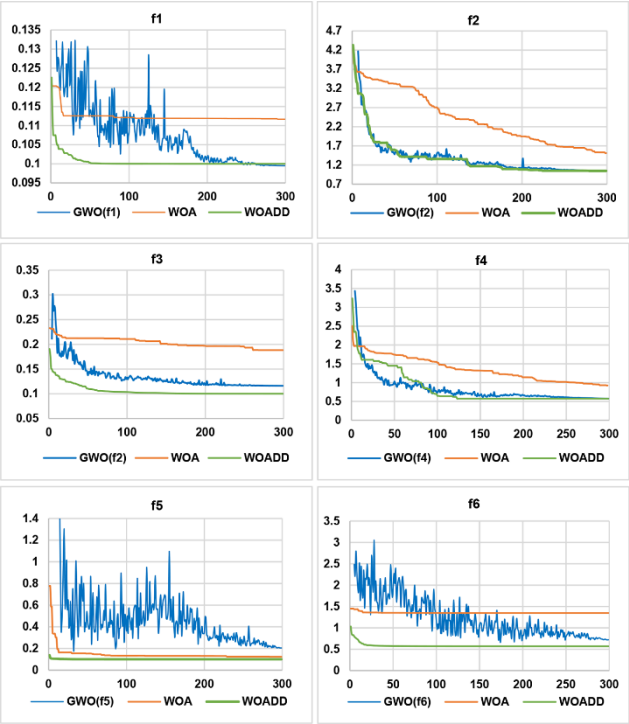

Fig. 2. Comparison of convergence curves of WOADD and literature algorithms obtained for the benchmark functions

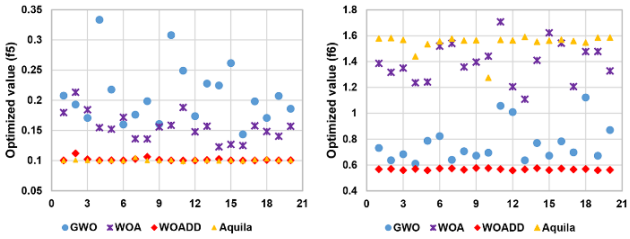

Fig. 3. The diversity of values produced by different algorithms in 20 runs for functions f_5 and f_6

VI. Conclusion

The paper presents an improved version of WOA that is specifically designed to handle dependent data. The proposed algorithm focuses on the valid range of data while searching the data space, leading to a more efficient search and faster attainment of the optimal solution. The experimental results demonstrate that this algorithm produces superior solutions, requiring fewer iterations to reach the optimal solution. Notably, for complex functions, the proposed algorithm achieves approximately 100% improvement over other tested algorithms. Additionally, the experiments show that the proposed algorithm's outcomes are more deterministic than other state-of-the-art ones, making it more suitable for applications that require precise outputs. Further development and refinement of such methods could unlock new insights and solutions from dependent data, an area that has received little attention thus far.

References

[1] Z. Lv, and R. Peng, "A novel meta-matching approach for ontology alignment using grasshopper optimization," *Knowledge-Based Systems*, vol. 201, pp. 106050, 2020.

[2] F. S. Gharehchopogh, and H. Gholizadeh, "A comprehensive survey: Whale Optimization Algorithm and its applications," *Swarm and Evolutionary Computation*, vol. 48, pp. 1-24, 2019.

[3] S. Mirjalili, S. M. Mirjalili, S. Saremi, and S. Mirjalili, "Whale optimization algorithm: theory, literature review, and application in designing photonic crystal filters," *Nature-Inspired Optimizers*, pp. 219-238, 2020.

[4] X. Xue, and J. Chen, "Optimizing sensor ontology alignment through compact co-firefly algorithm," *Sensors*, vol. 20, no. 7, pp. 2056, 2020.

[5] K. M. Ang, W. H. Lim, N. A. M. Isa, S. S. Tiang, and C. H. Wong, "A constrained multi-swarm particle swarm optimization without velocity for constrained optimization problems," *Expert Systems with Applications*, vol. 140, pp. 112882, Feb. 2020.

[6] F. A. Hashim, and A. G. Hussien, "Snake Optimizer: A novel meta-heuristic optimization algorithm," *Knowledge-Based Systems*, vol. 242, pp. 108320, 2022.

[7] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Computer methods in applied mechanics and engineering*, vol. 376, pp. 113609, 2021.

[8] I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, "Gradient-based optimizer: A new metaheuristic optimization algorithm," *Information Sciences*, vol. 540, pp. 131-159, 2020.

[9] M. Shehab, L. Abualigah, H. Al Hamad, H. Alabool, M. Alshinwan, and A. M. Khasawneh, "Moth–flame optimization algorithm: variants and applications," *Neural Computing and Applications*, vol. 32, pp. 9859-9884, 2020.

[10] L. Abualigah, D. Yousri, M. Abd Elaziz, A. A. Ewees, M. A. Al-Qaness, and A. H. Gandomi, "Aquila optimizer: a novel meta-heuristic optimization algorithm," *Computers & Industrial Engineering*, vol. 157, pp. 107250, 2021.

[11] S. Gholizadeh, M. Danesh, and C. Gheyratmand, "A new Newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames," *Computers & Structures*, vol. 234, pp. 106250, 2020.

[12] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46-61, 2014/03/01/, 2014.

[13] G. Sun, Y. Shang, K. Yuan, and H. Gao, "An Improved Whale Optimization Algorithm Based on Nonlinear Parameters and Feedback Mechanism," *International Journal of Computational Intelligence Systems*, vol. 15, no. 1, pp. 1-17, 2022.

[14] S. Mirjalili, and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1-14, 2013.

[15] W.-T. Pan, "A new fruit fly optimization algorithm: taking the financial distress model as an example," *Knowledge-Based Systems*, vol. 26, pp. 69-74, 2012.

[16] F. Zishan, E. Akbari, A. R. Sheikholeslami, and N. shafaghatian, "Optimization and Placement of DG Resources in the Network to Reduce Line Loading," *International Journal of Industrial Electronics Control and Optimization*, vol. 6, no. 2, pp. 89-100, 2023.

[17] N. Hashimzade, and M. A. Thornton, Handbook of research methods and applications in empirical microeconomics: Edward Elgar Publishing, 2021.

[18] D. Cicchella, M. Ambrosino, A. Gramazio, F. Coraggio, M. A. Musto, A. Caputi, D. Avagliano, and S. Albanese, "Using multivariate compositional data analysis (CoDA) and clustering to establish geochemical backgrounds in stream sediments of an onshore oil deposits area. The Agri River basin (Italy) case study," *Journal of Geochemical Exploration*, vol. 238, pp. 107012, 2022.

[19] G. Tepanosyan, L. Sahakyan, N. Maghakyan, and A. Saghatelyan, "Combination of compositional data analysis and machine learning approaches to identify sources and geochemical associations of potentially toxic elements in soil

and assess the associated human health risk in a mining city," *Environmental Pollution*, vol. 261, pp. 114210, 2020.
[20] H. Li, "Microbiome, metagenomics, and high-dimensional compositional data analysis," *Annual Review of Statistics and Its Application*, vol. 2, pp. 73-94, 2015.
[21] M. Greenacre, M. Martínez-Álvaro, and A. Blasco, "Compositional data analysis of microbiome and any-omics datasets: a validation of the additive logratio transformation," *Frontiers in microbiology*, vol. 12, pp. 727398, 2021.
[22] G. Coenders, and B. Ferrer-Rosell, "Compositional data analysis in tourism: review and future directions," *Tourism Analysis*, vol. 25, no. 1, pp. 153-168, 2020.
[23] L. Lalicic, E. Marine-Roig, B. Ferrer-Rosell, and E. Martin-Fuentes, "Destination image analytics for tourism design: An approach through Airbnb reviews," *Annals of Tourism Research*, vol. 86, pp. 103100, 2021.
[24] I. Janssen, A. E. Clarke, V. Carson, J. P. Chaput, L. M. Giangregorio, M. E. Kho, V. J. Poitras, R. Ross, T. J. Saunders, and A. Ross-White, "A systematic review of compositional data analysis studies examining associations between sleep, sedentary behaviour, and physical activity with health outcomes in adults," *Applied physiology, nutrition, and metabolism*, vol. 45, no. 10, pp. S248-S257, 2020.
[25] D. Dumuid, Ž. Pedišić, T. E. Stanford, J.-A. Martín-Fernández, K. Hron, C. A. Maher, L. K. Lewis, and T. Olds, "The compositional isotemporal substitution model: a method for estimating changes in a health outcome for reallocation of time between sleep, physical activity and sedentary behaviour," *Statistical methods in medical research*, vol. 28, no. 3, pp. 846-857, 2019.
[26] A. Alenazi, "A review of compositional data analysis and recent advances," *Communications in Statistics-Theory and Methods*, vol. 52, no. 16, pp. 5535-5567, 2023.
[27] D. Li, A. Srinivasan, Q. Chen, and L. Xue, "Robust covariance matrix estimation for high-dimensional compositional data with application to sales data analysis," *Journal of Business & Economic Statistics*, vol. 41, no. 4, pp. 1090-1100, 2023.
[28] T. Correa, M. Reyes, L. S. Taillie, C. Corvalán, and F. R. Dillman Carpentier, "Food advertising on television before and after a national unhealthy food marketing regulation in Chile, 2016–2017," *American journal of public health*, vol. 110, no. 7, pp. 1054-1059, 2020.
[29] T. P. Quinn, I. Erb, M. F. Richardson, and T. M. Crowley, "Understanding sequencing data as compositions: an outlook and review," *Bioinformatics*, vol. 34, no. 16, pp. 2870-2878, 2018.
[30] U. M. Khaire, and R. Dhanalakshmi, "Stability of feature selection algorithm: A review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1060-1073, 2022.
[31] D. Maulud, and A. M. Abdulazeez, "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, no. 4, pp. 140-147, 2020.
[32] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," *Artificial intelligence review*, vol. 55, no. 1, pp. 291-322, 2022.
[33] Y. Pan, Z. Su, A. Liu, W. Jingquan, N. Li, and Z. Xu, "A unified weight initialization paradigm for tensorial convolutional neural networks." pp. 17238-17257.
[34] C. Segura, C. A. C. Coello, G. Miranda, and C. León, "Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization," *Annals of Operations Research*, vol. 240, pp. 217-250, 2016.
[35] C. A. C. Coello, "Constraint-handling techniques used with evolutionary algorithms." In *Proceedings of the genetic and evolutionary computation conference companion*, pp. 1310-1333. 2022.
[36] G. Li, and Q. Zhang, "Multiple penalties and multiple local surrogates for expensive constrained optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 769-778, 2021.
[37] F. Samanipour, and J. Jelovica, "Adaptive repair method for constraint handling in multi-objective genetic algorithm based on relationship between constraints and variables," *Applied Soft Computing*, vol. 90, pp. 106143, 2020.
[38] W. Liu, K. Liu, and T. Deng, "Modelling, analysis and improvement of an integrated chance-constrained model for level of repair analysis and spare parts supply control," *International Journal of Production Research*, vol. 58, no. 10, pp. 3090-3109, 2020.
[39] B. Ji, X. Yuan, and Y. Yuan, "Modified NSGA-II for solving continuous berth allocation problem: Using multiobjective constraint-handling strategy," *IEEE transactions on cybernetics*, vol. 47, no. 9, pp. 2885-2895, 2017.
[40] B.-C. Wang, H.-X. Li, Q. Zhang, and Y. Wang, "Decomposition-based multiobjective optimization for constrained evolutionary optimization," *IEEE Transactions on systems, man, and cybernetics: systems*, vol. 51, no. 1, pp. 574-587, 2018.
[41] X. Li, S. Zeng, C. Li, and J. Ma, "Many-objective optimization with dynamic constraint handling for constrained optimization problems," *Soft Computing*, vol. 21, pp. 7435-7445, 2017.
[42] R. Jiao, S. Zeng, J. S. Alkasassbeh, and C. Li, "Dynamic multi-objective evolutionary algorithms for single-objective optimization," *Applied Soft Computing*, vol. 61, pp. 793-805, 2017.
[43] T. Xu, J. He, and C. Shang, "Helper and equivalent objectives: Efficient approach for constrained optimization," *IEEE transactions on cybernetics*, vol. 52, no. 1, pp. 240-251, 2020.
[44] S. Zeng, R. Jiao, C. Li, and R. Wang, "Constrained optimisation by solving equivalent dynamic loosely-constrained multiobjective optimisation problem," *International Journal of Bio-Inspired Computation*, vol. 13, no. 2, pp. 86-101, 2019.
[45] S. Mirjalili, and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, Vol. 95, pp. 51-67, 2016.
[46] V. Pawlowsky-Glahn, J. J. Egozcue, and R. Tolosana-Delgado, Modeling and analysis of compositional data, John Wiley & Sons, 2015.
[47] G.-Y. Ning, and D.-Q. Cao, "Improved whale optimization algorithm for solving constrained optimization problems," *Discrete Dynamics in Nature and Society*, vol. 2021, pp.1-13 2021.
[48] S. Mostafa Bozorgi, and S. Yazdani, "IWOA: An improved whale optimization algorithm for optimization problems," *Journal of Computational Design and Engineering*, vol. 6, no. 3, pp. 243-259, July 2019.
[49] J.-J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," In *Proceedings 2005 IEEE Swarm Intelligence Symposium*, pp. 68-75, 2005.

Asieh Ghanbarpour received her M.S. degree in Computer Engineering from Sharif University in 2010. She obtained her Ph.D. in Computer Engineering from the Iran University of Science and Technology in 2018. Currently, she is working as an Assistant Professor in the Computer Department at the University of Sistan and Baluchestan. Her research interests lie in Information Retrieval, Data Mining, Databases, and Graph Processing topics.

Soheil Zaremotlagh received his M.S. degree from the Tehran University in 2003 and his Ph.D. degree from the Amirkabir University of Technology in 2016. He joined the Faculty of Engineering at the University of Sistan and Baluchestan in 2006. His research interests primarily focus on applying artificial intelligence methods to earth science data.

Fahimeh Dabaghi Zarandi is currently an assistant professor in computer engineering department of Vali-e-Asr University of Rafsanjan. She received the Ph.D. degree in software engineering from Iran University of Science and Technology, Tehran, Iran, in September 2018. She received the M.S. Degree in software engineering from the Sharif University of Technology, Tehran, Iran, in August 2010. She received the BSc Degree in software engineering from Ferdowsi university of Mashhad, Iran, in August 2008. Her research interests lie in Green Communication, Data Mining, Graph Processing, and Internet of Things.

Appendix:

TABLE 1 DESCRIPTION OF BENCHMARK CONSTRAINED FUNCTIONS

Function	Improved function with penalty	Optimum solution
$f_1(x) = \sum_{i=0}^{n} x_i^2$ $s.j: \forall x_i: 0 \le x_i \le 1, \sum_{i=0}^{n} x_i = 1$	$f_1(x) = \sum_{i=1}^{n} x_i - \alpha \sum_{i=1}^{n} \max(0, x_i - 1) - \beta \sum_{i=1}^{n} \max(0, -x_i) - \gamma \left(1 - \sum_{i=0}^{n} x_i\right)$	$f_1(x^*) = \frac{1}{n}$ $x^* = (\frac{1}{n}, \dots, \frac{1}{n})$
$f_2(x) = \sum_{i=0}^{n} (i+1) \times x_i$ $s.j \begin{cases} \forall x_i: 0 \le x_i \le 1 \\ \sum_{i=0}^{n} x_i = 1 \end{cases}$	$f_2(x) = \sum_{i=1}^{n} (i+1) \times x_i - \alpha \sum_{i=1}^{n} \max(0, x_i - 1)$ $- \beta \sum_{i=1}^{n} \max(0, -x_i) - \gamma \left(1 - \sum_{i=0}^{n} x_i\right)$	$f_2(x^*) = 1$ $x^* = (1, 0, \dots, 0)$
$f_3(x) = \sum_{i=0}^{n} \frac{1}{i+1} \times x_i$ $s.j \begin{cases} \forall x_i: 0 \le x_i \le 1 \\ \sum_{i=0}^{n} x_i = 1 \end{cases}$	$f_3(x) = \alpha \sum_{i=1}^{n} \frac{1}{i+1} \times x_i - \alpha \sum_{i=1}^{n} \max(0, x_i - 1)$ $- \beta \sum_{i=1}^{n} \max(0, -x_i) - \gamma \left(1 - \sum_{i=0}^{n} x_i\right)$	$f_3(x^*) = 1$ $x^* = (1, 0, \dots, 0)$
$f_4(x) = \sum_{i=0}^{n} \frac{(i+1)^2}{i+2} \times x_i$ $s.j \begin{cases} \forall x_i: 0 \le x_i \le 1 \\ \sum_{i=0}^{n} x_i = 1 \end{cases}$	$f_4(x) = \sum_{i=0}^{n} \frac{(i+1)^2}{i+2} \times x_i - \alpha \sum_{i=1}^{n} \max(0, x_i - 1)$ $- \beta \sum_{i=1}^{n} \max(0, -x_i) - \gamma \left(1 - \sum_{i=0}^{n} x_i\right)$	$f_4(x^*) = \frac{1}{2}$ $x^* = (1, 0, \dots, 0)$
$f_5(x) = \sum_{i=0}^{m} x_i^2 + \sum_{i=m+1}^{n} x_i^2$ $s.j \begin{cases} \forall x_i: 0 \le x_i \le 1 \\ \sum_{i=0}^{m} x_i = c \\ \sum_{i=m+1}^{n} x_i = 1-c \end{cases}$	$f_5(x) = \alpha \left(\sum_{i=0}^{m} x_i + \sum_{i=m+1}^{n} x_i \right) - \beta \left(\left\lvert \frac{1}{2} - \sum_{i=0}^{m} x_i \right\rvert + \left\lvert \frac{1}{2} - \sum_{i=m+1}^{n} x_i \right\rvert \right) - \gamma \left(\sum_{i=1}^{n} \max(0, -x_i) + \sum_{i=1}^{n} \max(0, x_i - 1) \right)$	$f_5(x^*) = \frac{1}{m} + \frac{1}{n-m}$ $x^* = (\frac{1}{m}, \dots, \frac{1}{n-m}, \dots, \frac{1}{n-m})$
$f_6(x) = \sum_{i=0}^{m} (i+1) \times x_i + \sum_{i=m+1}^{n} \frac{1}{i+1} \times x_i$ $s.j \begin{cases} \forall x_i: 0 \le x_i \le 1, \sum_{i=0}^{m} x_i = c \\ \sum_{i=m+1}^{n} x_i = 1-c \end{cases}$	$f_6(x) = \alpha \left(\sum_{i=0}^{m} (i+1) \times x_i + \sum_{i=m+1}^{n} \frac{1}{i+1} \times x_i \right)$ $- \beta \left(\left\lvert c - \sum_{i=0}^{m} x_i \right\rvert + \left\lvert 1-c - \sum_{i=m+1}^{n} x_i \right\rvert \right)$ $- \gamma \left(\sum_{i=1}^{n} \max(0, -x_i) + \sum_{i=1}^{n} \max(0, x_i - 1) \right)$	$f_6(x^*) = \frac{1}{m} + \frac{1}{n-m}$ $x^* = (c, 0, \dots, 0, 1-c)$

TABLE 2 COMPARISON OF OPTIMIZATION RESULTS OBTAINED CONSTRAINED OPTIMIZATION FUNCTIONS

		n = 5			n = 10		
Function	Algorithm	opt	avg	std	opt	avg	std
f_1	GWO	**0.19801980**	0.19802	3.81E-09	**0.09951292**	0.09951	3.29E-06
	WOA	0.24697830	0.76064	0.473464	0.11257629	0.24801	0.136155
	Aquila	0.20901384	0.21568	0.049482	0.1011492	0.1102	0.03038
	WOADD	0.20000125	0.23054	0.025687	0.10013077	0.10685	0.014231
f_2	GWO	1.01336514	1.01336	0.000061	1.07093814	1.07101	4.58E-05
	WOA	1.08175004	4.45874	4.393632	1.49786942	2.51204	0.784425
	Aquila	2.54880186	2.91290	0.282923	5.49780352	5.40050	0.300903
	WOADD	**1.00046018**	1.34106	0.537621	**1.04863548**	2.14027	1.335185
f_3	GWO	0.21012547	0.20951	1.90E-04	0.11002846	0.11002	7.64E-05
	WOA	0.24386565	0.19516	0.025801	0.16962024	0.16114	0.008623
	Aquila	0.32208246	0.42317	0.08324	0.29050187	0.29944	0.050842
	WOADD	**0.20402433**	0.23263	0.051775	**0.10004526**	0.10433	0.016416
f_4	GWO	0.52344523	0.52378	5.12E-04	0.54897454	0.54874	6.76E-04
	WOA	0.52788077	1.46388	0.641348	0.92521517	1.76456	0.695900
	Aquila	1.68200241	2.16610	0.311962	4.42580623	4.63140	0.451971
	WOADD	**0.50056879**	0.81622	0.609542	**0.54862247**	3.64583	1.978977
f_5	GWO	0.25597429	0.25578	0.00025	0.20839877	0.21728	0.000266
	WOA	0.28083862	0.33046	0.08898	0.15356303	0.17039	0.017592
	Aquila	0.27418764	0.25381	0.082323	0.10035238	0.09692	0.00682
	WOADD	**0.20967541**	0.25214	0.03853	**0.10046314**	0.12067	0.019322
f_6	GWO	0.72157598	0.72146	5.43E-04	0.71640018	0.71629	4.53E-04
	WOA	0.88517801	0.86809	0.052462	1.25271829	1.23933	0.035963
	Aquila	0.94366280	1.02790	0.232031	1.56251892	1.5288	0.158072
	WOADD	**0.60162363**	0.73374	0.242773	**0.56118408**	1.00936	0.490113

TABLE 3 COMPARISON OF OPTIMIZATION RESULTS OBTAINED FOR f_5 AND f_6 BASED ON DIFFERENT VALUES OF C

		c = 0.1			c = 0.3			c = 0.5		
Fun	Algorithm	opt	avg	std	opt	avg	std	opt	avg	std
f_5	GWO	0.21	0.21	3.47E-04	0.21	0.21	3.3E-04	0.21	0.22	0.0002
	WOA	0.29	0.16	0.0204	0.17	0.17	0.0137	0.15	0.17	0.0176
	Aquila	0.39	0.38	0.2089	0.85	0.19	0.0175	0.10	0.09	0.0068
	WOADD	**0.16**	0.19	0.0334	**0.12**	0.14	0.0238	**0.10**	0.12	0.0193
f_6	GWO	0.34	0.34	1.59E-04	0.62	0.61	3.62E-04	0.72	0.72	4.5E-04
	WOA	0.46	0.46	0.013092	0.88	0.86	0.034454	1.25	1.24	0.0359
	Aquila	0.49	0.73	0.34441	0.92	0.94	0.41512	1.56	1.52	0.1580
	WOADD	**0.20**	0.28	0.098156	**0.38**	0.80	0.37041	**0.56**	1.01	0.4901