JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER

p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

MSE-CDO: A Multi-Strategy Enhanced Cloud Drift Optimizer for Global and Constrained Engineering Optimization

Aziz Nasser Boraik[1*], Hesham Bahamish[2] , Saeed Awadh Al-Shami[3]

[1,3]Faculty of Computing and Engineering, Information Technology, University of Science and Technology, Aden, Yemen
[2]College of Computers and Information Technology, Computer Sciences, Hadhramout University, Mukalla, Yemen
Email: [1*]aziz.bin.sama@gmail.com, [2]bahamish@hu.edu.ye, [3]shami377@gmail.com
(*Email Corresponding Author: aziz.bin.sama@gmail.com)
Received: August 17, 2026 | Revision: September 2, 2026 | Accepted: September 2, 2026

Abstract

Cloud Drift Optimization (CDO) is a recent nature-inspired metaheuristic with a simple adaptive search framework. Despite its adaptive mechanisms, CDO does not explicitly regulate initial space coverage, population dispersion during evolution, or agent-specific local refinement. This study proposes a Multi-Strategy Enhanced Cloud Drift Optimizer, termed MSE-CDO. First, randomized multistart maximin Latin hypercube sampling is employed to improve the spatial distribution of the initial population. Second, adaptive Gaussian mutation is introduced after the original CDO update to generate controlled perturbations according to the population dispersion and search stage, while greedy selection retains better candidates. Third, a directional pattern search mechanism is incorporated to strengthen local refinement through individual search directions and adaptive step lengths. The proposed algorithm is evaluated on unimodal and multimodal benchmark functions and constrained engineering design problems and is compared with the original CDO and several established metaheuristic algorithms. The experimental results show that MSE-CDO improves the performance of the original CDO on most benchmark functions, achieves the best overall Friedman rank among the compared algorithms, and obtains competitive solutions for the considered engineering problems. These results indicate that the proposed strategies improve the search and convergence performance of CDO while preserving its original framework.

Keywords : Cloud Drift Optimization, MSE-CDO, metaheuristic optimization, Latin hypercube sampling, Gaussian mutation, pattern search, engineering optimization.

1. INTRODUCTION

In recent decades, optimization problems have been encountered across a broad range of practical domains [1]–[4]. These domains include structural engineering, energy management, manufacturing, transportation, and computational science. The purpose of an optimization problem is to determine the best values of a set of decision variables by minimizing or maximizing an objective function under given constraints [1]. For simple problems, traditional mathematical optimization methods can provide accurate solutions. However, many real-world problems are nonlinear, nonconvex, multimodal, high-dimensional, or constrained, and the derivative information of the objective function may be difficult or impossible to obtain [1], [3], [5]. Under these conditions, traditional optimization methods may become difficult to apply. Therefore, metaheuristic algorithms have been widely studied as alternative methods for solving complex optimization problems [1]–[6].

Population-based metaheuristics can be interpreted through the interaction between exploration and exploitation. Exploration promotes coverage of distinct regions, whereas exploitation concentrates evaluations around promising solutions. If the search concentrates too early, the population may converge prematurely; if diversification remains excessive, convergence may be unnecessarily slow. Effective search therefore requires a balance that depends on the problem structure and the current search stage.

Existing metaheuristics employ diverse update mechanisms derived from biological interaction, social learning, evolutionary operators, and physical processes. Particle Swarm Optimization uses individual and social learning to guide the movement of particles [7]. Grey Wolf Optimizer simulates the leadership hierarchy and hunting behavior of grey wolves [8], while Whale Optimization Algorithm models the encircling and bubble-net feeding behavior of whales [9]. Harris Hawks Optimization uses several attacking strategies according to the escaping energy of prey [10], and Marine Predators Algorithm employs different movement mechanisms during different search stages [11]. Metaheuristic algorithms have also been applied outside conventional numerical optimization. For example, modified population-based algorithms have been used for protein structure prediction and multiple sequence alignment [12], [13]. These studies show that metaheuristic algorithms can be adapted to different optimization problems through suitable search mechanisms.

According to the No Free Lunch theorem, no optimization algorithm can achieve the best performance for all optimization problems [14]. As optimization problems become more complex, improving existing algorithms remains an important research direction. One common approach is to introduce several complementary strategies into an existing optimizer. Different strategies can act on different stages of the optimization process. For example, pattern search and population renewal have been used to improve the Tree Seed Algorithm [16]. Latin hypercube sampling has been introduced to improve the initial population distribution of RIME [17], while dynamic Gaussian mutation has been used to strengthen the search ability of the Aquila Optimizer [18]. Hybridization has also been applied to discrete optimization

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

problems [15]. Together, these studies motivate examining initialization, perturbation, and local refinement as distinct intervention points within an otherwise unchanged optimizer.

Cloud Drift Optimization (CDO) is a recently proposed nature-inspired metaheuristic that simulates the movement of cloud particles under atmospheric drift forces [19]. In CDO, candidate solutions are represented by cloud particles. Fitness-ranked adaptive weights and nonlinear control functions are used to regulate the search process. According to a probabilistic rule, a cloud performs either an exploitation movement around the current best solution or a broader exploratory movement. In addition, the algorithm includes a decreasing probability of random reinitialization to reduce stagnation. The original CDO was evaluated on unimodal and multimodal benchmark functions and constrained engineering design problems and showed competitive optimization performance [19].

Three characteristics of the original CDO motivate the present modification. First, uniform random initialization provides no explicit space-filling criterion, so a finite population may contain clusters and poorly represented regions. Second, as the search proceeds and individuals move toward promising regions, the population may become increasingly concentrated. The random relocation mechanism of CDO can introduce new solutions, but it is not directly related to the current population dispersion. Finally, local refinement is mainly performed through the original stochastic position-updating mechanism. Each cloud does not maintain an individual search direction and step length for subsequent local refinement. These three properties motivate targeted changes at initialization, post-update diversification, and local refinement.

To improve the original CDO, we propose a Multi-Strategy Enhanced Cloud Drift Optimizer, called MSE-CDO. MSE-CDO modifies three stages of CDO while retaining the original drift equations. Randomized multistart maximin Latin hypercube sampling is first used instead of ordinary random initialization to obtain a more distributed initial population. Adaptive Gaussian mutation is then introduced after the original CDO position update. The mutation scale is adjusted according to the population dispersion and the current search stage. Greedy selection is used to retain the better solution. Finally, a directional pattern search strategy is added to improve the local search ability of CDO. Each cloud has its own search direction and step length, which are adjusted during the search process.

The main contributions of this paper are summarized as follows:

(1) To improve the distribution of the initial population, a randomized multistart maximin Latin hypercube sampling strategy is introduced into the initialization phase of CDO.

(2) To maintain population diversity during the search, an adaptive Gaussian mutation strategy is added after the original CDO position update. Its mutation scale is adjusted according to the population dispersion and search progress, and greedy selection is used to retain better solutions.

(3) To improve the local search ability of CDO, a directional pattern search strategy is introduced. Each cloud maintains an individual search direction and step length during the optimization process.

(4) To evaluate the performance of the proposed MSE-CDO, it is tested on unimodal and multimodal benchmark functions and constrained engineering design problems and compared with the original CDO and several established metaheuristic algorithms.

The rest of this paper is structured as follows. Section 2 reviews related studies on metaheuristic optimization and improvement strategies. Section 3 describes the proposed MSE-CDO in detail. Section 3 presents the experimental results on benchmark functions and constrained engineering design problems. Finally, the last Section summarizes this paper and discusses future work.

2. LITERATURE REVIEW

Metaheuristic algorithms have been widely investigated as derivative free methods for numerical and engineering optimization. Although these algorithms differ in their inspiration and position updating rules, most of them begin with a population of candidate solutions and improve this population through repeated stochastic movements [1], [6]. Their performance is affected by the distribution of the initial solutions, the ability to explore different regions, and the accuracy of the final search. According to the No Free Lunch theorem, no algorithm can provide the best performance for every optimization problem [14]. This result encourages researchers to develop new methods and improve existing algorithms for different classes of problems.

Metaheuristic algorithms can be roughly divided into swarm intelligence inspired, human inspired, evolutionary inspired, and natural like inspired classes according to the origin of their search mechanisms [20]. The swarm intelligence class models the collective behaviour of biological populations and generally allows search agents to exchange information or follow selected leaders. Particle Swarm Optimization represents each agent by a position and velocity and directs its movement using the best position previously obtained by the agent and the best position found by the population [7]. Grey Wolf Optimizer simulates the leadership hierarchy and hunting mechanism of grey wolves, where the three leading solutions guide the remaining agents during encircling and hunting [8]. Other algorithms in this class include Whale Optimization Algorithm [9], Harris Hawks Optimization [10], Marine Predators Algorithm [11], Dragonfly Algorithm [21], Grasshopper Optimization Algorithm [22], Seagull Optimization Algorithm [23], Chimp Optimization Algorithm [24], and FOX [25]. These methods reproduce different forms of hunting, migration, interaction, or foraging to determine the movement of the population. The human inspired class, in contrast, develops its search process from human learning and social activities. Human Learning Optimization uses individual learning, social learning, random

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

exploration learning, and relearning operators to generate binary solutions and search for improved alternatives [26]. Teaching Learning Based Optimization considers the population as a class of learners. Its teacher phase moves learners according to the influence of the best learner, whereas its learner phase improves solutions through interaction between randomly selected learners [27].

The evolutionary inspired class models inheritance, variation, and survival. Genetic Algorithm applies selection, crossover, and mutation to evolve candidate solutions [28], whereas Differential Evolution generates donor vectors from scaled differences between population members and retains improved trials through selection [29]. The natural like inspired class models physical or mathematical processes. Snow Ablation Optimizer represents sublimation and melting to support exploration and exploitation [30], while Fick's Law Algorithm simulates diffusion from higher to lower concentration regions [31].

The application of metaheuristics is not restricted to conventional numerical optimization. Their search mechanisms have also been adapted to computational biology, where the number of possible solutions can be very large. Bahamish et al. applied the Bat Algorithm to protein conformational search [32]. The same research area was further investigated by modifying the Crow Search Algorithm through genetic crossover, Metropolis acceptance, and a new solution generation procedure for protein structure prediction [12]. Hussein et al. also applied the basic Flower Pollination Algorithm to protein multiple sequence alignment [13]. These studies show that population-based algorithms can be adapted to domain specific problems when the solution representation and fitness evaluation are properly defined.

Another direction in the literature concerns the use of several compatible strategies within one optimizer. Liu et al. [16] proposed an improved Tree Seed Algorithm using pattern search, dimension permutation, and an elimination update mechanism. Pattern search was used to strengthen solution detection, while the remaining strategies-maintained diversity and renewed inferior solutions. Zhong et al. [17] strengthened RIME by replacing random initialization with Latin hypercube sampling and adding a distance-based selection procedure. This method produced a more distributed initial population and considered the spatial distance among candidate solutions. Zeng et al. [18] introduced nonlinear control, spiral movement, and dynamic Gaussian mutation into the Aquila Optimizer. Gaussian and mixed mutation schemes were also used in other algorithms. Duan et al. [33] combined manta ray foraging and Gaussian mutation with elephant herding optimization, while Nautiyal et al. [34] incorporated mutation schemes into the Salp Swarm Algorithm. Chen and Zhu [35] combined comprehensive learning with Cauchy and Gaussian mutation in the Escape Algorithm. These studies indicate that initialization, mutation, population renewal, and local search perform different functions during optimization. Their effectiveness depends on their position within the algorithm and the rule used to accept newly generated solutions. CDO was introduced as a population-based optimizer inspired by the movement of clouds under atmospheric drift forces [19]. The original algorithm uses random initialization, adaptive weights obtained from fitness ranking, nonlinear control functions, probabilistic exploration and exploitation movements, and a decreasing probability of random relocation. Its performance was evaluated using eleven numerical functions and two constrained engineering problems. However, random initialization does not ensure a well distributed population, and the original drift process does not maintain an individual local search direction. Population diversity may also decrease when agents move toward similar promising regions. Therefore, the original method leaves room for improving initial coverage, preserving dispersion, and increasing final search accuracy.

Recent studies have started to use improved forms of CDO in different applications. An improved CDO was combined with a two-layer model predictive control method for economic dispatch in microgrids [36]. Bao et al. [37] used an improved CDO to optimize a support vector machine for transformer fault diagnosis. Other CDO based methods were reported for underwater acoustic signal processing [38], regional energy management [39], and treatment recommendation [40]. These studies indicate that CDO can be modified for particular applications. However, among the CDO studies reviewed in this work, no variant was identified that space filling initialization, adaptive Gaussian mutation, and individual directional pattern search for both global numerical optimization and constrained engineering design.

The present study addresses this gap through MSE-CDO. Randomized multistart maximin Latin hypercube sampling is used to improve the initial population. Adaptive Gaussian mutation introduces controlled variation after the original CDO update, while greedy selection prevents unsuccessful perturbations from replacing the current solutions. Directional pattern search then examines forward and opposite regions using an individual direction and step size for each cloud. This arrangement retains the original CDO movement and adds three search mechanisms with separate functions: initial population distribution, diversity preservation, and local refinement.

3. MULTI-STRATEGY ENHANCED CLOUD DRIFT OPTIMIZER (MSE-CDO)

This section defines the three auxiliary operators added to CDO and specifies their execution order. The original drift update is retained unchanged: multistart maximin Latin hypercube sampling is applied during initialization, adaptive Gaussian mutation follows the CDO position update, and the accepted candidate is subsequently processed by the directional refinement stage.

3.1 Randomized Multistart Maximin Latin Hypercube Initialization

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

The original population initialization procedure in CDO was replaced with a randomized multistart maximin Latin hypercube strategy following the approach adopted by Costa [41]. For a population of N solutions in a D dimensional search space, the interval of each decision variable is divided into N equally sized strata. One value is sampled randomly from each stratum, and the resulting values are independently permuted across dimensions to construct a Latin hypercube design. This procedure is repeated K times to generate K independent candidate designs. In the present study, K was set to 8. Each candidate is first generated in the normalized domain $[0,1]^D$ and is subsequently transformed to the actual search bounds using the following expression:

$$X = L + Z \odot (U - L) \quad (1)$$

where Z denotes the normalized Latin hypercube design, while L and U represent the lower and upper bound vectors, respectively.

The most widely dispersed candidate design is selected according to the maximin criterion. For each candidate $Z(k)$, the minimum Euclidean distance between any two distinct solutions is calculated as

$$\delta_k = \min_{i \neq j} \|z_i^{(k)} - z_j^{(k)}\|_2 \quad (2)$$

The design associated with the largest value of δk is then used as the initial population, according to

$$k^* = \arg\max_k \delta_k \quad (3)$$

This procedure promotes a more uniform spatial distribution and reduces clustering among the initial solutions without using objective function information during initialization. It also avoids the deterministic inclusion of the center of the search space. The selected population is supplied directly to the CDO search process without modifying its original position update equations.

3.2 Adaptive Gaussian mutation strategy

As the optimization process continues, the population gradually moves toward promising regions of the search space. Although this behavior improves convergence, the reduction in population diversity may restrict the movement of individuals and increase the possibility of stagnation around a local optimum. To address this limitation, an adaptive Gaussian mutation strategy is introduced after the original Cloud Drift Optimization position update and before the directional pattern search stage. The strategy generates a perturbed candidate for each active individual while excluding individuals that have already undergone random relocation.

The Gaussian candidate is generated as follows:

$$X_{i,j}^G(t+1) = X_{i,j}^C(t+1) + M_{i,j}(t)\sigma_j(t)Z_{i,j} \quad (4)$$

$$Z_{i,j} \sim \mathcal{N}(0,1) \quad (5)$$

where $X_{i,j}^C(t+1)$ represents the position produced by the original Cloud Drift Optimization update, $Z_{i,j}$ is a random value sampled from a standard Gaussian distribution, and $M_{i,j}(t)$ is a mutation mask. Each dimension is selected with probability $1/D$, where D denotes the problem dimension. If no dimension is selected, one coordinate is randomly activated to ensure that every active individual receives a valid perturbation.

The adaptive mutation scale is defined as follows:

$$\sigma_j(t) = \frac{b(t)}{\sqrt{D}} \max\left(S_j(t), \sqrt{\varepsilon} R_j\right) \quad (6)$$

where $S_j(t)$ denotes the standard deviation of the current population in dimension j, R_j represents the width of the corresponding search interval, and ε is the numerical precision. The coefficient $b(t)$ gradually decreases during the optimization process. The Gaussian scale coefficient is defined as b(t)=1−t/T, where t and T denote the current and maximum numbers of iterations, respectively. The strategy permits relatively broad perturbations during the early iterations and produces finer local variations as the population approaches the final stage.

After boundary control, the Gaussian candidate is compared with the original updated position. The selection rule is expressed as follows:

$$X_i(t+1) = \begin{cases} X_i^G(t+1), & f\left(X_i^G(t+1)\right) < f\left(X_i^C(t+1)\right) \\ X_i^C(t+1), & \text{otherwise} \end{cases} \quad (7)$$

This greedy selection preserves the better candidate and prevents unsuccessful perturbations from replacing the current solution. The accepted population is then passed to the directional pattern search stage for further refinement.

3.3 Adaptive Directional Refinement Based on Pattern Search

In the original CDO, the search agents update their positions according to the best solution and randomly selected individuals. During the optimization process, the movement obtained in the current iteration is not used to determine the search direction in the next iteration. Because no agent-specific directional state is retained, the current displacement does not directly inform the direction examined in a subsequent local refinement step. The pattern search method is a direct search method that does not require derivative information. Its search process includes positive direction movement, negative direction movement, and step length adjustment. Based on this idea, an agent-specific directional search is applied after the stochastic CDO update to provide an additional derivative-free refinement step.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

After the original CDO position update is completed, the obtained position of the ith agent is denoted by $\tilde{X}_i^{t+1}$. Another agent $X_{q_i}^t$ is randomly selected from the population, where $q_i \neq i$. According to the search direction of the current agent, the pattern search position is generated by

$$Y_{i,j}^{t+1} = \tilde{X}_{i,j}^{t+1} + \text{rand} \times S_i^t \times \left(\tilde{X}_{i,j}^{t+1} - X_{q_i,j}^t\right) \quad (8)$$

$$Y_{i,j}^{t+1} = \tilde{X}_{i,j}^{t+1} - \text{rand} \times S_i^t \times \left(\tilde{X}_{i,j}^{t+1} - X_{q_i,j}^t\right) \quad (9)$$

where $Y_{i,j}^{t+1}$ represents the jth dimension of the pattern search position, rand is a uniformly distributed random number in the interval $(0,1)$, and S_i^t is the step length of the ith agent. Equation (8) is used for positive direction search, while Eq. (9) is used for negative direction search. The generated position is restricted within the lower and upper bounds before its fitness value is calculated.

A judgment flag jud_i is used to determine the search direction of each agent. The initial value of jud_i is set to 0, and the initial step length is set to $S_i^0 = 2$. When $jud_i = 0$, the agent searches in the positive direction. When $jud_i = 1$, the agent searches in the negative direction. The pattern search position is compared with the position generated by the original CDO update according to

$$X_i^{t+1} = \begin{cases} Y_i^{t+1}, & f(Y_i^{t+1}) < f\left(\tilde{X}_i^{t+1}\right), \\ \tilde{X}_i^{t+1}, & \text{otherwise.} \end{cases} \quad (10)$$

If the pattern search position has a better fitness value, it replaces the original updated position, and jud_i is set to 0. If the pattern search position does not improve the fitness value, the original updated position is retained, jud_i is set to 1, and the step length is reduced according to

$$S_i^{t+1} = S_i^t \times k \quad (11)$$

where k is a constant smaller than 1, and its value is set to 0.97. Each agent has an independent judgment flag and step length. By changing the search direction after an unsuccessful movement, the agent can examine the opposite region. During the iterative process, the step length is gradually reduced, which increases the local search ability in the later stage and improves the convergence accuracy of CDO.

3.4 Computational Complexity Analysis

Let N denote the population size, D the problem dimension, T the maximum number of iterations, and K the number of Latin hypercube candidates. The original CDO has a computational complexity of O(TND) [19]. In MSE-CDO, the multistart maximin Latin hypercube initialization requires $O(KN^2D)$ due to the pairwise distance calculations. The CDO update, adaptive Gaussian mutation, and directional pattern search operate on the population during the iterative process and retain a complexity of O(TND). Therefore, the overall computational complexity of MSE-CDO is

$$O(KN^2D + TND) \quad (12)$$

Since K is fixed in this study, it can be simplified to

$$O(N^2D + TND) \quad (13)$$

Thus, MSE-CDO introduces an additional one-time computational cost during initialization, while the order of the iterative complexity remains the same as that of the original CDO.

4. RESULTS AND DISCUSSION

In this section we evaluate MSE-CDO against the original CDO and nine established algorithms using eleven benchmark functions and two constrained engineering design problems.

4.1 Mathematical Benchmark Functions

Eleven mathematical benchmark functions are employed to evaluate the performance of MSE-CDO. The same functions, dimensionality, population size, and iteration limit used in the original CDO study are adopted to align the benchmark protocol. These functions include unimodal and multimodal problems with different search ranges, while the dimension of each function is set to 100. Their definitions, search ranges, and global optimum values are presented in Table 1.

The benchmark set contains seven unimodal functions (F1–F7) and four multimodal functions (F8–F11). Unimodal functions mainly examine the exploitation and convergence accuracy of an algorithm since they contain a single global optimum. The multimodal functions contain several local optima and are therefore useful for evaluating exploration and

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

the ability to avoid local solutions. The set also includes difficult characteristics such as the narrow valley of Rosenbrock, noise in the Quartic function, and highly multimodal landscapes in Rastrigin and the penalized functions.

Table 1. Benchmark Functions (F1–F11)

Type	Benchmark name	Function	Dim	Range	fmin
Unimodal	Sphere	$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	100	[−100, 100]	0
	Schwefel 2.22	$f_2(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	100	[−10, 10]	0
	Schwefel 1.2 (reconstructed)	$f_3(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	100	[−100, 100]	0
	Schwefel 2.21	$f_4(\mathbf{x}) = \max\{\lvert x_i \rvert : 1 \le i \le n\}$	100	[−100, 100]	0
	Rosenbrock	$f_5(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	100	[−30, 30]	0
	Step	$f_6(\mathbf{x}) = \sum_{i=1}^{n} \text{floor}\,(x_i + 0.5)^2$	100	[−100, 100]	0
	Quartic with Noise	$f_7(\mathbf{x}) = \sum_{i=1}^{n} i\, x_i^4 + r, \quad r \sim U(0,1)$	100	[−1.28, 1.28]	0
Multimodal	Rastrigin	$f_8(\mathbf{x}) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	100	[−5.12, 5.12]	0
	Ackley	$f_9(\mathbf{x}) = -20\exp\left(-0.2 \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$	100	[−32, 32]	0
	Generalized Penalized 1	$f_{10}(\mathbf{x}) = \frac{\pi}{n}\left\{ 10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = k(x_i - a)^m, \quad x_i > a$ $u(x_i, a, k, m) = 0, \quad -a \le x_i \le a$ $u(x_i, a, k, m) = k(-x_i - a)^m, \quad x_i < -a$	100	[−50, 50]	0
	Generalized Penalized 2	$f_{11}(\mathbf{x}) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	100	[−50, 50]	0

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

For all benchmark functions, the population size is set to 1,000 search agents and the maximum number of iterations is 200. Each function is independently solved 30 times, following the experimental settings of the original CDO [19]. In the present study, MSE-CDO and the original CDO were executed under these conditions. For ChOA, DA, FOX, GWO, GOA, HHO, MPA, SOA, and PSO, the numerical results reported in the original CDO paper [19] were directly adopted since the same benchmark functions and experimental settings are used. This allows the proposed MSE-CDO to be compared with CDO and the nine algorithms under a consistent experimental basis. The original CDO study explicitly reports 1,000 search agents, 200 iterations, 30 independent runs, and the same nine comparison algorithms.

4.1.1 Results of benchmark functions

Table 2 presents the comparison between MSE-CDO and the original CDO based on the mean results of the independent runs. MSE-CDO reports lower mean objective values than CDO on nine of the eleven functions, while F8 and F9 result in ties at the reported precision. On F2 and F4, MSE-CDO reaches the global optimum, while CDO obtains values close to zero. On F1 and F3, both algorithms reach values close to machine precision, but MSE-CDO obtains the lower mean values. For F5, F6, and F7, MSE-CDO improves the mean result of CDO by 1.44%, 18.61%, and 77.07%, respectively. On F10 and F11, the improvement reaches 38.06% and 62.12%, respectively. These results indicate that the introduced strategies effectively improve the optimization performance of the original CDO.

Table 3 presents the comparison results of MSE-CDO and the other algorithms. For the unimodal functions F1–F7, MSE-CDO obtains the best average result on F1, F2, F3, F4, and F7. HHO obtains the best average result on F5 and F6. However, MSE-CDO performs better than the original CDO on these two functions, and it also obtains the best single solution on F6. The standard deviation results show that MSE-CDO has stable performance on these functions. In particular, zero standard deviation is obtained on F1–F4.

For the multimodal functions, MSE-CDO reaches the global optimum on F8, with the same result obtained by CDO, ChOA, and HHO. On F9, MSE-CDO and CDO obtain the best average result among the compared algorithms. MSE-CDO does not rank first on F10 and F11, where HHO and MPA obtain the best average results, respectively. However, MSE-CDO obtains clearly better results than the original CDO on both functions. This shows that the proposed modifications improve the ability of CDO to deal with multimodal search spaces, although some comparison algorithms perform better on particular functions.

Figure 1 shows the convergence curves of MSE-CDO and CDO on F4, F7, and F11. On F4, the fitness value of MSE-CDO decreases continuously during the optimization process and reaches the global optimum, while CDO remains at a higher fitness level for a large part of the iterations before decreasing in the later stage. On F7, both algorithms decrease rapidly in the initial iterations, but MSE-CDO maintains a lower average fitness value and obtains a better final solution. For F11, the two algorithms have similar behavior in the early search stage, while MSE-CDO obtains lower fitness values in the later iterations and reaches a better final result. Therefore, the convergence curves show that MSE-CDO improves the convergence behavior and final search accuracy of the original CDO on the selected functions.

According to the Friedman ranking results in Table 4, MSE-CDO obtains the lowest average rank and ranks first among the eleven algorithms. HHO ranks second, followed by the original CDO. It can be seen that MSE-CDO has the best overall ranking on the benchmark functions. The results further verify that the proposed improvement strategies enhance the overall optimization performance of CDO.

Table 2. Performance Improvement of MSE-CDO over the Original CDO

Function	MSE-CDO Mean	CDO Mean	Interpretation
F1	7.0651E-322	1.2055E-321	MSE-CDO better; both near machine precision
F2	0	9.35E-297	MSE-CDO reaches 0
F3	6.8675E-322	1.3339E-321	MSE-CDO better; both near machine precision
F4	0	2.89E-159	MSE-CDO reaches 0
F5	92.9444	94.2996	1.44% improvement
F6	0.001191432	0.001463856	18.61% improvement
F7	1.66E-06	7.24E-06	77.07% improvement
F8	0	0	Tie
F9	4.44E-16	4.44E-16	Tie
F10	0.133095	0.214881	38.06% improvement
F11	0.022652	0.059795	62.12% improvement

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER

p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

Table 3. Comparison of Optimization Performance of Algorithms. Bold indicates the best value.

Function		MSE-CDO	CDO	ChOA	DA	FOX	GWO	GOA	HHO	MPA	SOA	PSO
F1	Best	0	9.8813e-324	2.5942E-23	6832.9649	2.102E-54	1.079E-27	11773.2075	1.7064E-45	0.0001086612378	14.2606	0.0001545
	Aver	7.0651e-322	1.2055e-321	1.5E-22	7500.12	3.1E-53	5.4E-26	12500.5	2.1E-44	0.00021034	16.872	0.00024
	Std	0	0	1.1E-21	1300.6	5.2E-52	8.2E-25	2100.3	3.4E-43	0.00007612	4.28	0.00009
F2	Best	0	2.09E-299	2.6045E-16	49.2886	3.7119E-52	8.5163E-17	88.1418	5.9421E-24	0.002738614188	0.59128	0.044772
	Aver	0	9.35E-297	4.3E-15	53.4123	6.1E-51	9.7E-16	91.452	6.5E-23	0.004315	0.8734	0.06123
	Std	0	0	2.1E-14	10.126	2.9E-50	4.2E-15	14.278	1.2E-22	0.001128	0.3612	0.0156
F3	Best	0	0	2.2304E-07	149588.4542	170.20	5.2676E-06	32258.9053	8.5073E-36	1018.781061	10797.0246	68.5558
	Aver	6.8675e-322	1.3339e-321	3.4E-06	155000.32	190.45	7.8E-05	33000.6	9.1E-35	1032.17	11000.43	75.234
	Std	0	0	1.1E-05	20000.7	55.32	1.2E-04	7000.12	3.4E-34	234.5	1500.3	12.543
F4	Best	0	7.58E-299	2.9805E-06	58.6074	20.1045	7.6592E-07	22.1783	1.2456E-24	0.0195175541	91.1506	1.1346
	Aver	0	2.89E-159	4.7E-05	63.214	24.5	9.8E-06	25.742	1.9E-23	0.028612	100.5	1.5623
	Std	0	1.53E-158	1.2E-04	8.954	5.812	3.4E-05	6.218	7.2E-23	0.00678	21.45	0.6213
F5	Best	92.59706609	94.05190283	132.922	28316948.8289	86.5500	27.1946	2626954.3765	0.50579	98.46515378	1469.6574	139.5597
	Aver	92.94438561	94.29960792	140.45	29000000.54	90.12	32.89	2750000.3	0.60421	102.21	1500.24	145.67
	Std	0.178522803	0.139528942	12.3	2500000.6	13.98	6.5	300000.45	0.0987	20.8	145.3	17.3
F6	Best	0.000771463	0.001067953	1.0086	3020.2748	0.11678	0.7408	12221.7733	0.00090438	8.70555924	35.1585	7.5646E-03
	Aver	0.001191432	0.001463856	1.4512	3200.45	0.13214	0.8214	12500.87	0.001132	9.5214	40.123	9.2314E-03
	Std	0.000164193	0.000191086	0.3124	500.72	0.02134	0.154	1300.42	0.000213	2.154	5.978	2.311E-03
F7	Best	4.57E-08	3.03E-07	0.000450	14.1567	0.19808	0.0014876	0.71128	0.0011455	0.006622951612	0.079796	0.18263
	Aver	1.66E-06	7.24E-06	0.000612	25.4831	0.26549	0.003214	1.0243	0.002348	0.008937	0.1254	0.23187
	Std	1.50E-06	5.91E-06	0.00089	5.8742	0.07312	0.000978	0.3294	0.000756	0.002317	0.04287	0.08421
F8	Best	0	0	0	963.6891	0	3.6314	526.3356	0	1.942421045E-05	163.0408	64.747
	Aver	0	0	0	1024.872	0.1523	4.8572	608.4321	0	2.3461	194.237	79.654
	Std	0	0	0	72.9843	0.0283	1.3278	85.2734	0	0.8795	23.8723	10.4782

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

F9	Best	4.44E-16	4.44E-16	2.524E-13	14.7042	1.2305	9.743E-14	11.7128	4.4409E-13	0.0005013256751	19.9657	0.3812
	Aver	4.44E-16	4.44E-16	4.896E-14	18.7351	1.6784	2.384E-13	14.3562	1.258E-12	0.0009372	24.3214	0.5732
	Std	0	0	1.782E-14	3.1256	0.5783	9.521E-14	2.4683	5.847E-13	0.0002148	6.7923	0.1487
F10	Best	0.097656925	0.160418482	0.05607	10777380.3006	9.6587	0.05592	53.0676	4.1182E-07	4.947763972E-05	1.0647	2.2667E-06
	Aver	0.133095225	0.214881133	0.07345	1.23E+08	12.9842	0.07987	67.8543	0.000005684	6.3412E-04	1.5783	3.1482E-04
	Std	0.026218099	0.035926184	0.01237	1.85E+07	2.4578	0.01823	14.3572	0.00001254	1.4738E-04	0.4265	7.581E-05
F11	Best	0.002221765	0.016457528	1.04980	0.54258	0.1387	0.56801	0.024964	0.00019915	1.422116818E-09	2.4863	0.013164
	Aver	0.022652476	0.059795151	1.2874	0.67924	0.1982	0.74321	0.035147	0.0003147	1.8573E-08	3.0478	0.017436
	Std	0.015441541	0.029938232	0.2736	0.11287	0.0459	0.12876	0.009587	0.0000732	4.938E-08	0.9823	0.003479

Table 4. Friedman test rankings on Mathematical Benchmark Functions

Algorithm	Average Rank	Overall Rank
MSE-CDO	**2.27**	1
HHO	2.77	2
CDO	3.18	3
GWO	5.18	4
ChOA	5.23	5
MPA	5.82	6
FOX	6.09	7
PSO	6.27	8
SOA	9.27	9
GOA	9.64	10
DA	10.27	11

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

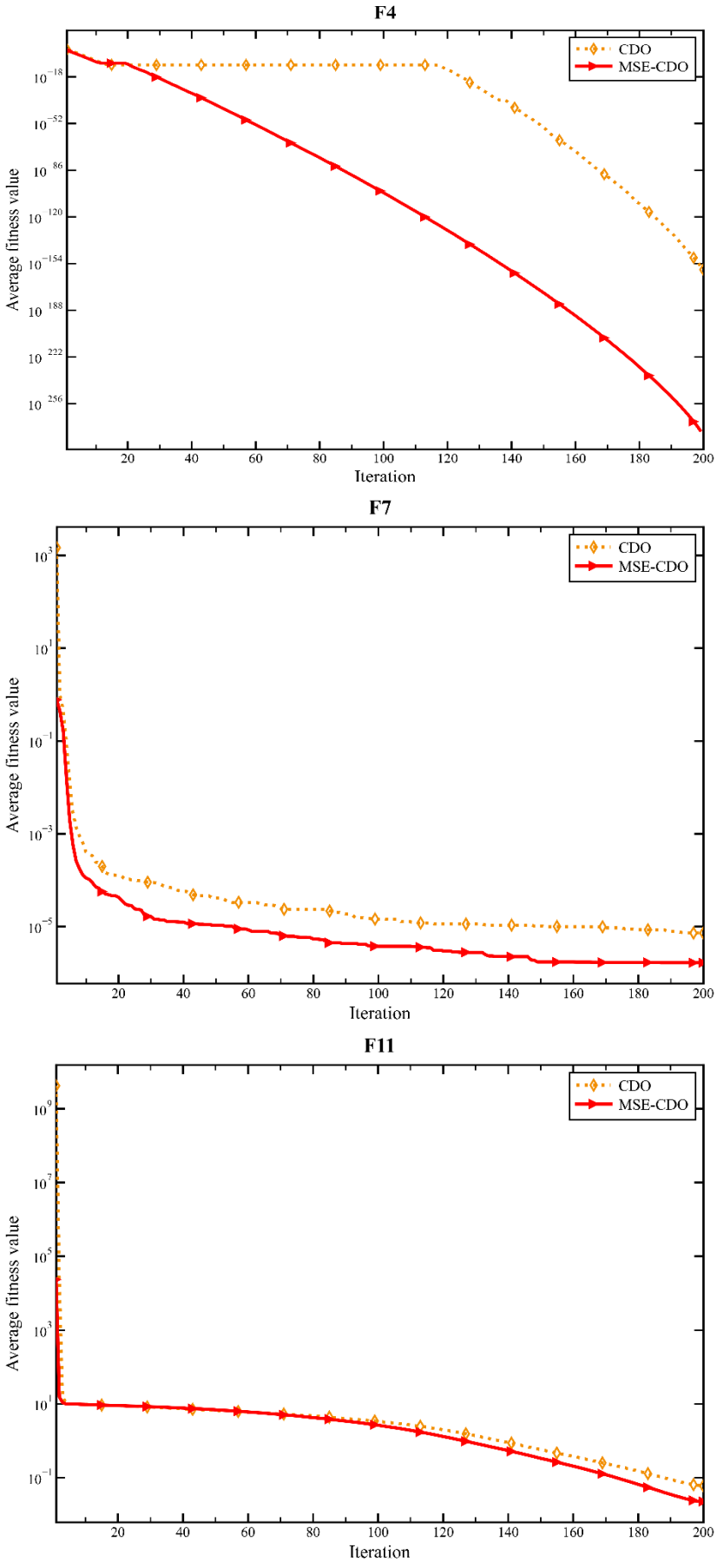

Figure 1. Convergence of MSE-CDO and CDO on the test functions (F4, F7, and F11)

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

4.2 Results on Constrained Engineering Design Problems

This section evaluates the performance of MSE-CDO in solving constrained engineering optimization problems. Two practical engineering design problems are considered, including the Three-Bar Truss Design and the Tension/Compression Spring Design. These problems contain different design variables and nonlinear constraints and are used to test the performance of the proposed algorithm in engineering optimization.

In our experiments, we run the proposed MSE-CDO and the original CDO [19] on the two engineering problems. The population size and the maximum number of iterations are set to 30 and 500, respectively, which are the same settings used for the comparative algorithms reported in [42]. For comparison, we use the optimization results of MSINGO [42], NGO [43], DE [29], SSA [44], WOA [9], DBO [45], POA [46], SOA [23], and BWO [47] reported in [42]. Only MSE-CDO and CDO are executed in this study. The results for the Three-Bar Truss Design and Tension/Compression Spring Design are presented in the following subsections.

4.2.1 Three-Bar Truss Design

The Three-Bar Truss Design problem is an optimization problem in civil engineering where the objective is to minimize the volume of the three-bar truss under the given constraints. The schematic representation of the problem is shown in Figure 2. It has two design variables, namely x1 and x2, representing the cross-sectional areas of the truss members. The mathematical formulation of the problem is defined as follows [42]:

$$Consider: x = [x_1\ x_2] \quad (14)$$

$$Minimize: f(x) = (2\sqrt{2}x_1 + x_2)l \quad (15)$$

Subject to:

$$g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2x_1^2 + 2x_1x_2}}P - \sigma \leq 0 \quad (16)$$

$$g_2(x) = \frac{x_2}{\sqrt{2x_1^2 + 2x_1x_2}}P - \sigma \leq 0 \quad (17)$$

$$g_3(x) = \frac{1}{\sqrt{2x_2 + x_1}}P - \sigma \leq 0 \quad (18)$$

$$Parameter\ range: 0 \leq x_1, x_2 \leq 1$$

where l = 100 cm, P = 2 kN/cm², and σ represents the allowable stress.

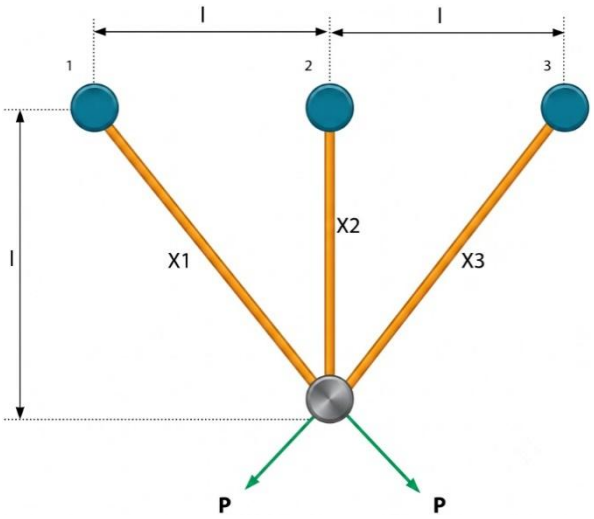

Figure 2. Schematic representation of the Three-Bar Truss Design problem.

Table 5 summarizes the reported Three-Bar Truss solutions. DE obtains the best optimal value of 2.6389584×10² and ranks first. MSE-CDO obtains 2.6389585×10² with x1 = 0.788725863 and x2 = 0.408104831, while CDO obtains the same optimal value with x1 = 0.788797619 and x2 = 0.407901964. Both MSE-CDO and CDO rank second. MSE-CDO also obtains a better optimal value than MSINGO, NGO, SSA, WOA, DBO, POA, SOA, and BWO.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

Table 5. Three-Bar Truss Design

Algorithm	x1	x2	Optimal Value	Rank
MSE-CDO	0.788725863	0.408104831	2.6389585E+02	2
CDO	0.788797619	0.407901964	2.6389585E+02	2
MSINGO	0.78866778	0.4082692	2.6389586E+02	3
NGO	0.78871489	0.40813758	2.6389602E+02	4
DE	0.78867514	0.40824829	2.6389584E+02	1
SSA	0.78884713	0.41517572	2.6463723E+02	8
WOA	0.82522025	0.36892016	2.7029955E+02	10
DBO	0.79085504	0.40224642	2.6391223E+02	6
POA	0.78794119	0.41042386	2.6390581E+02	5
SOA	0.79690303	0.41399512	2.6679773E+02	9
BWO	0.78871128	0.40914414	2.6399565E+02	7

4.2.2 Tension/Compression Spring Design

The Tension/Compression Spring Design problem is an engineering optimization problem to minimize the weight of a tension/compression spring while satisfying the constraint conditions. The schematic representation of the problem is shown in Figure 3. There are three variables that require optimization: spring wire diameter d, spring coil diameter D, and the number of active coils p. The mathematical formula of the Tension/Compression Spring Design problem is defined as follows [42]:

$$\text{Consider: } \mathrm{x} = [x_1, x_2, x_3] = [d, D, p] \tag{19}$$

$$\text{Minimize: } f(\mathrm{x}) = (x_3 + 2)x_2x_1^2 \tag{20}$$

Subject to:

$$g_1(\mathrm{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \tag{21}$$

$$g_2(\mathrm{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \tag{22}$$

$$g_3(\mathrm{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \tag{23}$$

$$g_4(\mathrm{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{24}$$

$$0.05 \leq x_1 \leq 2, \qquad 0.25 \leq x_2 \leq 1.3, \qquad 2 \leq x_3 \leq 15$$

Table 6 presents the optimization results for the Tension/Compression Spring Design problem. MSE-CDO obtains the best optimal value of 1.266623×10^{-2} and ranks first among all the algorithms. CDO obtains an optimal value of 1.267054×10^{-2} and ranks second. The optimal variables obtained by MSE-CDO are $d = 0.051866089$, $D = 0.360984723$, and $p = 11.04342255$, while CDO obtains $d = 0.05223004$, $D = 0.369872185$, and $p = 10.55748274$. These results show that MSE-CDO improves the optimal solution obtained by the original CDO and also outperforms MSINGO, NGO, DE, SSA, WOA, DBO, POA, SOA, and BWO on this engineering problem.

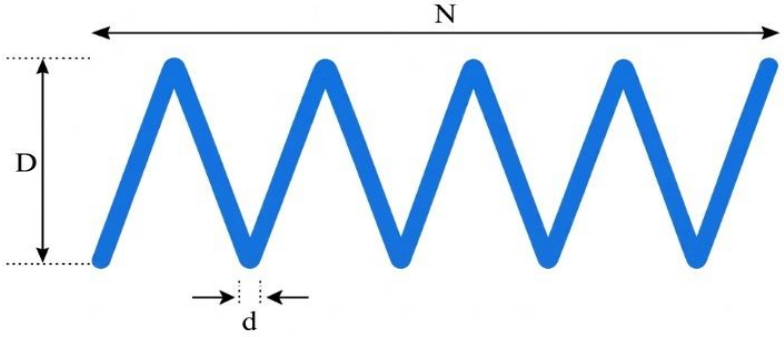

Figure 3. Schematic representation of the Tension/Compression Spring Design problem

Table 6. Tension/Compression Spring Design

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

Algorithm	D	D	p	Optimal Value	Rank
MSE-CDO	0.051866089	0.360984723	11.04342255	1.266623E-02	1
CDO	0.05223004	0.369872185	10.55748274	1.267054E-02	2
MSINGO	0.05166107	0.35617546	11.36666667	1.267369E-02	3
NGO	0.05187427	0.36129583	11.06666667	1.267450E-02	4
DE	0.05181379	0.35957038	11.16666667	1.268436E-02	5
SSA	0.05060498	0.32639958	13.86666667	1.314612E-02	8
WOA	0.05220499	0.36931603	11.36666667	1.298871E-02	7
DBO	0.05541281	0.4763829	10.30000000	1.408497E-02	10
POA	0.05203765	0.05203765	11.03333333	1.275111E-02	6
SOA	0.05826311	0.53740755	6.03333333	1.412553E-02	11
BWO	0.05109965	0.34071944	13.56666667	1.325690E-02	9

5. CONCLUSION

In this paper, we propose a Multi-Strategy Enhanced Cloud Drift Optimizer named MSE-CDO. Firstly, randomized multistart maximin Latin hypercube sampling is used instead of random initialization to improve the distribution of the initial population. Secondly, adaptive Gaussian mutation is introduced after the original CDO position update to maintain population diversity and reduce the possibility of stagnation during the search process. Finally, directional pattern search is added to strengthen the local search ability and improve the convergence accuracy of CDO. We evaluated the proposed MSE-CDO on unimodal and multimodal benchmark functions and compared it with the original CDO and several well-known metaheuristic algorithms. MSE-CDO achieved lower mean objective values than CDO on nine of the eleven benchmark functions, with ties on the remaining two. We also found that MSE-CDO achieved the best overall Friedman ranking when compared with the original CDO and nine other established metaheuristic algorithms. These results indicate that the introduced strategies improve the exploitation ability, exploration ability, and convergence performance of CDO without losing the advantages of its original search process. Additionally, we applied MSE-CDO to constrained engineering design problems to verify its performance in practical optimization. In the Three-Bar Truss Design problem, MSE-CDO achieves a competitive solution that is close to the best solution among the comparison algorithms. In the Tension/Compression Spring Design problem, MSE-CDO outperforms the original CDO and the other comparison algorithms. The results of the engineering problems demonstrate that MSE-CDO has good capability in dealing with constrained engineering optimization problems.

In the future, we will verify MSE-CDO on more complex and large-scale optimization problems and study its performance in different practical applications. We will also investigate hybrid methods to further improve the performance of the algorithm, such as combining it with other local search or gradient-based methods.

REFERENCES

[1] E. H. Houssein, M. K. Saeed, G. Hu, and M. M. Al-Sayed, “Metaheuristics for solving global and engineering optimization problems: Review, applications, open issues and challenges,” *Arch. Comput. Methods Eng.*, vol. 31, pp. 4485–4519, 2024, doi: https://doi.org/10.1007/s11831-024-10168-6.

[2] K. Rajwar, K. Deep, and S. Das, “An exhaustive review of metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges,” *Artif. Intell. Rev.*, vol. 56, pp. 13187–13257, 2023, doi: https://doi.org/10.1007/s10462-023-10470-y.

[3] R. Salgotra, P. Sharma, S. Raju, and A. H. Gandomi, “A contemporary systematic review on meta-heuristic optimization algorithms with their MATLAB and Python code reference,” *Arch. Comput. Methods Eng.*, vol. 31, pp. 1749–1822, 2024, doi: https://doi.org/10.1007/s11831-023-10030-1.

[4] E. H. Cui, Z. Zhang, C. J. Chen, and W. K. Wong, “Applications of nature-inspired metaheuristic algorithms for tackling optimization problems across disciplines,” *Sci. Rep.*, vol. 14, Art. no. 9403, 2024, doi: https://doi.org/10.1038/s41598-024-56670-6.

[5] M. S. Shaikh *et al.*, “Applications, classifications, and challenges: A comprehensive evaluation of recently developed metaheuristics for search and analysis,” *Artif. Intell. Rev.*, vol. 58, Art. no. 390, 2025, doi: https://doi.org/10.1007/s10462-025-11377-6.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

[6] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, 2013, doi: https://doi.org/10.1016/j.ins.2013.02.041.

[7] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci. (MHS)*, Nagoya, Japan, 1995, pp. 39–43, doi: https://doi.org/10.1109/MHS.1995.494215.

[8] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014, doi: https://doi.org/10.1016/j.advengsoft.2013.12.007.

[9] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[10] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, 2019, doi: https://doi.org/10.1016/j.future.2019.02.028.

[11] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine Predators Algorithm: A nature-inspired metaheuristic," *Expert Syst. Appl.*, vol. 152, Art. no. 113377, 2020, doi: https://doi.org/10.1016/j.eswa.2020.113377.

[12] H. A. Bahamish, N. M. Al-Aidroos, and A. N. Boraik, "Modified Crow Search Algorithm for protein structure prediction," *Int. Res. J. Eng. Technol.*, vol. 8, no. 2, pp. 348–354, Feb. 2021.

[13] A. M. Hussein, R. Abdullah, N. AbdulRashid, and A. N. B. Ali, "Protein multiple sequence alignment by basic flower pollination algorithm," in *Proc. 2017 8th Int. Conf. Inf. Technol. (ICIT)*, 2017, pp. 833–838, doi: https://doi.org/10.1109/ICITECH.2017.8079955.

[14] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997, doi: https://doi.org/10.1109/4235.585893.

[15] A. F. J. Al-Gburi, S. Naim, and A. N. Boraik, "Hybridization of Bat and Genetic Algorithm to solve the N-queens problem," *Bull. Electr. Eng. Inform.*, vol. 7, no. 4, pp. 626–632, Dec. 2018, doi: https://doi.org/10.11591/eei.v7i4.1351.

[16] J. Liu, Y. Hou, Y. Li, and H. Zhou, "A multi-strategy improved tree–seed algorithm for numerical optimization and engineering optimization problems," *Sci. Rep.*, vol. 13, Art. no. 10768, 2023, doi: https://doi.org/10.1038/s41598-023-37958-5.

[17] R. Zhong, J. Yu, C. Zhang, and M. Munetomo, "SRIME: A strengthened RIME with Latin hypercube sampling and embedded distance-based selection for engineering optimization problems," *Neural Comput. Appl.*, vol. 36, no. 12, pp. 6721–6740, 2024, doi: https://doi.org/10.1007/s00521-024-09424-4.

[18] L. Zeng, M. Li, J. Shi, and S. Wang, "Spiral Aquila Optimizer based on dynamic Gaussian mutation: Applications in global optimization and engineering," *Neural Process. Lett.*, vol. 55, pp. 11653–11699, 2023, doi: 10.1007/s11063-023-11394-y.

[19] M. Alibabaei Shahraki, "Cloud drift optimization algorithm as a nature-inspired metaheuristic," *Discov. Comput.*, vol. 28, Art. no. 173, 2025, doi: https://doi.org/10.1007/s10791-025-09671-6.

[20] G. Hu, Y. Guo, W. Zhao, and E. H. Houssein, "An adaptive snow ablation-inspired particle swarm optimization with its application in geometric optimization," *Artif. Intell. Rev.*, vol. 57, Art. no. 332, 2024, doi: https://doi.org/10.1007/s10462-024-10946-5.

[21] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, 2016, doi: https://doi.org/10.1007/s00521-015-1920-1.

[22] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper Optimisation Algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, 2017, doi: https://doi.org/10.1016/j.advengsoft.2017.01.004.

[23] G. Dhiman and V. Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," *Knowl.-Based Syst.*, vol. 165, pp. 169–196, 2019, doi: https://doi.org/10.1016/j.knosys.2018.11.024.

[24] M. Khishe and M. R. Mosavi, "Chimp optimization algorithm," *Expert Syst. Appl.*, vol. 149, Art. no. 113338, 2020, doi: https://doi.org/10.1016/j.eswa.2020.113338.

[25] H. Mohammed and T. Rashid, "FOX: A FOX-inspired optimization algorithm," *Appl. Intell.*, vol. 53, no. 1, pp. 1030–1050, 2023, doi: https://doi.org/10.1007/s10489-022-03533-0.

[26] L. Wang, R. Yang, H. Ni, W. Ye, M. Fei, and P. M. Pardalos, "A human learning optimization algorithm and its application to multi-dimensional knapsack problems," *Applied Soft Computing*, vol. 34, pp. 736–743, 2015, doi: https://doi.org/10.1016/j.asoc.2015.06.004.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

[27] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning based optimization: A novel method for constrained mechanical design optimization problems," *Comput. Aided Des.*, vol. 43, no. 3, pp. 303–315, 2011, doi: https://doi.org/10.1016/j.cad.2010.12.015.

[28] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[29] R. Storn and K. Price, "Differential Evolution–A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997, doi: https://doi.org/10.1023/A:1008202821328.

[30] L. Deng and S. Liu, "Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design," *Expert Syst. Appl.*, vol. 225, Art. no. 120069, 2023, doi: https://doi.org/10.1016/j.eswa.2023.120069.

[31] F. A. Hashim, R. R. Mostafa, A. G. Hussien, S. Mirjalili, and K. M. Sallam, "Fick's Law Algorithm: A physical law based algorithm for numerical optimization," *Knowl. Based Syst.*, vol. 260, Art. no. 110146, 2023, doi: https://doi.org/10.1016/j.knosys.2022.110146.

[32] H. A. Bahamish, N. M. Al-Aidroos, and A. N. Boraik, "Bat Algorithm for protein conformational search," in *Proc. 2019 1st Int. Conf. Intell. Comput. Eng. (ICOICE)*, Hadhramout, Yemen, 2019, pp. 1-7, doi: https://doi.org/10.1109/ICOICE48418.2019.9035182.

[33] Y. Duan, C. Liu, S. Li, X. Guo, and C. Yang, "Manta ray foraging and Gaussian mutation-based elephant herding optimization for global optimization," *Eng. Comput.*, vol. 39, pp. 1085–1125, 2023, doi: https://doi.org/10.1007/s00366-021-01494-5.

[34] B. Nautiyal, R. Prakash, V. Vimal, G. Liang, and H. Chen, "Improved Salp Swarm Algorithm with mutation schemes for solving global optimization and engineering problems," *Eng. Comput.*, vol. 38, Suppl. 5, pp. S3927–S3949, 2022, doi: https://doi.org/10.1007/s00366-020-01252-z.

[35] H. Chen and X. Zhu, "An enhanced escape algorithm with comprehensive learning and Cauchy–Gaussian mutation for reservoir optimization," *Sci. Rep.*, vol. 16, Art. no. 16410, 2026, doi: https://doi.org/10.1038/s41598-026-46087-8.

[36] P. He, C. Ma, D. Zhou, and S. M. Muyeen, "An improved Cloud Drift Optimization based two-layer MPC for economic dispatch of microgrids with hybrid energy storage," in *Proc. 2025 7th Int. Conf. Smart Power Internet Energy Syst. (SPIES)*, 2025, pp. 259–264, doi: https://doi.org/10.1109/SPIES67451.2025.11381524.

[37] W. Bao, X. Lin, H. Jiang, J. Xiao, and R. Lin, "Transformer fault diagnosis based on support vector machine optimized by improved Cloud Drift Optimization," in *Proc. 2025 9th Int. Conf. Electr., Mech. Comput. Eng. (ICEMCE)*, 2025, pp. 515–519, doi: https://doi.org/10.1109/ICEMCE68156.2025.11466692.

[38] G. Li, Y. Xie, and H. Yang, "Noise reduction method of underwater acoustic signal based on trend perception optimization and double-level classification," *Measurement*, vol. 281, Art. no. 121944, 2026, doi: https://doi.org/10.1016/j.measurement.2026.121944.

[39] Z. Xiong, L. Han, J. Xu, X. Li, and Y. Xu, "Enhancing power quality in regional integrated energy systems: A novel multidimensional flexible management strategy for high proportions of renewable energy integration," *J. Energy Eng.*, vol. 152, no. 4, Art. no. 04026043, 2026, doi: https://doi.org/10.1061/JLEED9.EYENG-6727.

[40] R. Suseendra, T. Tamilvizhi, and R. Surendran, "Hybrid Stacked Autoencoder-Graph Attention Network with Chebyshev Cloud Drift Optimization for Treatment Recommendation," in *Proc. 2nd International Conference on Visual Analytics and Data Visualization (ICVADV 2026)*, 2026, pp. 681–687, doi : https://doi.org/10.1109/ICVADV67766.2026.11470461.

[41] A. Costa, E. Di Buccio, M. Melucci, and G. Nannicini, "Efficient parameter estimation for information retrieval using black-box optimization," IEEE Trans. Knowl. Data Eng., vol. 30, no. 7, pp. 1240–1253, 2018, doi: https://doi.org/10.1109/TKDE.2017.2761749.

[42] H. Liu, J. Xiao, Y. Yao, S. Zhu, Y. Chen, R. Zhou, Y. Ma, M. Wang, and K. Zhang, "A Multi-Strategy Improved Northern Goshawk Optimization Algorithm for Optimizing Engineering Problems," *Biomimetics*, vol. 9, no. 9, Art. no. 561, 2024, doi: https://doi.org/10.3390/biomimetics9090561.

[43] M. Dehghani, S. Hubalovsky, and P. Trojovsky, "Northern Goshawk Optimization: A New Swarm-Based Algorithm for Solving Optimization Problems," *IEEE Access*, vol. 9, pp. 162059–162080, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3133286.

JURNAL KOMPUTER TEKNOLOGI INFORMASI SISTEM KOMPUTER
p-ISSN: 2962-3022 | e-ISSN: 2963-7104
Volume 5 No 2 September 2026
Page: 2064-2079
http://ejurnal.lkpkaryaprima.id/index.php/juktisi
DOI: https://doi.org/10.62712/juktisi.v5i2.1530

[44] J. Xue and B. Shen, "A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm," *Systems Science & Control Engineering*, vol. 8, no. 1, pp. 22–34, 2020, doi: https://doi.org/10.1080/21642583.2019.1708830.

[45] J. Xue and B. Shen, "Dung Beetle Optimizer: A New Meta-Heuristic Algorithm for Global Optimization," *The Journal of Supercomputing*, vol. 79, pp. 7305–7336, 2023, doi: https://doi.org/10.1007/s11227-022-04959-6.

[46] P. Trojovský and M. Dehghani, "Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications," *Sensors*, vol. 22, no. 3, Art. no. 855, 2022, doi: https://doi.org/10.3390/s22030855.

[47] C. Zhong, G. Li, and Z. Meng, "Beluga Whale Optimization: A Novel Nature-Inspired Metaheuristic Algorithm," *Knowledge-Based Systems*, vol. 251, Art. no. 109215, 2022, doi: https://doi.org/10.1016/j.knosys.2022.109215.