*Robotica* (2025), 1–38
doi:10.1017/S0263574725000049

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

# Self-adaptive differential evolution-based coati optimization algorithm for multi-robot path planning

Lun Zhu[1], Guo Zhou[2], Yongquan Zhou[1,3], Qifang Luo[1,3], Huajuan Huang[1,3] and Xiuxi Wei[1,3]

[1]College of Artificial Intelligence, Guangxi Minzu University, Nanning, China
[2]Department of Science and Technology Teaching, China University of Political Science and Law, Beijing, China
[3]Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning, China
**Corresponding author:** Yongquan Zhou; Email: zhouyongquan@gxun.edu.cn

**Received:** 23 July 2024; **Revised:** 2 December 2024; **Accepted:** 6 January 2025

**Keywords:** differential evolution; coati optimization algorithm; self-adaptive differential evolution-based coati optimization; multi-robot path planning; metaheuristic

## Abstract

The multi-robot path planning problem is an NP-hard problem. The coati optimization algorithm (COA) is a novel metaheuristic algorithm and has been successfully applied in many fields. To solve multi-robot path planning optimization problems, we embed two differential evolution (DE) strategies into COA, a self-adaptive differential evolution-based coati optimization algorithm (SDECOA) is proposed. Among these strategies, the proposed algorithm adaptively selects more suitable strategies for different problems, effectively balancing global and local search capabilities. To validate the algorithm's effectiveness, we tested it on CEC2020 benchmark functions and 48 CEC2020 real-world constrained optimization problems. In the latter's experiments, the algorithm proposed in this paper achieved the best overall results compared to the top five algorithms that won in the CEC2020 competition. Finally, we applied SDECOA to optimization multi-robot online path planning problem. Facing extreme environments with multiple static and dynamic obstacles of varying sizes, the SDECOA algorithm consistently outperformed some classical and state-of-the-art algorithms. Compared to DE and COA, the proposed algorithm achieved an average improvement of 46% and 50%, respectively. Through extensive experimental testing, it was confirmed that our proposed algorithm is highly competitive. The source code of the algorithm is accessible at: https://ww2.mathworks.cn/matlabcentral/fileexchange/164876-HDECOA.

## 1. Introduction

Robot systems, primarily accomplishing desired tasks through close collaboration between individual or multiple robots and humans, have now become an indispensable part of the development in agriculture, industry, and service sectors. In this trend of development, robot path planning has become one of the most crucial issues in robot systems. Multi-robot path planning is an NP-hard problem, which describes the scenario where, in the same working environment, these robots need to move from their initial positions to their respective destinations at the minimum cost while avoiding collisions with other robots or obstacles during the movement. Multi-robot path planning involves multiple constraints, particularly in dynamically changing environments. Previous work has proposed several solutions to avoid collisions and coordinate between multiple robots [1]. Classical methods for solving robot path planning problems typically include potential field techniques [2], bounding box representations [3], Breadth-First Search and Depth-First Search [4], A* algorithm [5], and exploring random tree methods [6]. However, these methods often fail to effectively address multi-robot path planning as the scale of robots or obstacles increases. Current researchers primarily utilize artificial neural networks and metaheuristic algorithms to

© The Author(s), 2025. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

solve such problems. Metaheuristic algorithms primarily mimic biological behaviors or natural phenomena as heuristic methods and are primarily categorized into four types. The first type is population-based metaheuristic algorithms, exemplified by particle swarm optimization (PSO) [7], which simulates the foraging behavior of bird flocks, and ant colony optimization (ACO) [8], which mimics the process where ant populations release pheromones along their paths during foraging to find optimal routes based on pheromone concentration. Similar algorithms include artificial bee colony (ABC) [9]. The second type is evolutionary-based metaheuristic algorithms, such as genetic algorithm (GA) [10] and differential evolution (DE) [11], which emulate the process of biological evolution where the most optimal genetic individuals are preserved over generations. The third type is based on chemical and physical phenomena, grounded in theoretical foundations, with representative algorithms including gravitational search algorithm (GSA) [12] and Archimedes optimization algorithm (AOA) [13]. The fourth type is inspired by human social behaviors and can draw inspiration from various aspects of daily life. Examples include lungs performance-based optimization (LPO) [14], which simulates human lung function activity, and football team training algorithm (FTTA) [15], which mimics the training process of a football team. However, due to the homogeneity of their update strategies, these algorithms tend to fall into local optima to some degree when solving problems, and their parameter settings are relatively uniform, making them unsuitable for solving diverse problems.

This article is a study on path planning problem based on metaheuristic algorithm. According to the No Free Lunch theorem, no single algorithm can solve all problems optimally. Therefore, researchers often integrate multiple strategies and algorithms to improve the performance of path planning algorithms. Geng et al. [16] combined sparrow search algorithm (SSA) with reverse learning strategy to solve a robot path planning problem on a grid map. Parhi et al. [17] merged the classical method of linear regression (LR) with GSA called RGSA, incorporating multiple chaos strategies to obtain optimal paths in multi-humanoids (Nao robots) movement problems. Li et al. [18] proposed a hybrid algorithm based on Improved GA and Dynamic Window Approach for solving mobile robot path planning problems. Xu et al. [19] combined a new fourth-order Bezier transition curve with an improved PSO algorithm, proposed a novel method for smooth path planning of mobile robots. Nazarahari et al. [20] proposed an innovative artificial potential field (APF) algorithm to find all feasible paths between start and end points in a discrete grid environment and developed an Enhanced GA to improve initial path and find the optimal path between start and end points in robot path planning problems. Zhang et al. [21] proposed a hybrid algorithm of GA and firefly algorithm to enhance the responsiveness and computational capability of mobile robots during movement. Dai et al. [22] introduced an Improved ACO utilizing characteristics of A* algorithm and MAX-MIN ant system to achieve efficient search capability for mobile robot path planning in complex maps. Chen et al. [23] addressed path planning problems for mobile robots in known environments, proposed a grid-based hybrid APF and ACO path planning method. Zhang et al. [24] proposed a turning point-based grey wolf optimizer (GWO) for solving the path planning problem of patrol robots. The algorithm utilizes the concepts of roulette wheel selection and crossover mutation to broaden the search scope of the initial population. Additionally, the convergence factor function varies with the number of obstacles, enhancing the performance of the proposed algorithm. Liu et al. [25] integrated the A* algorithm with an improved GWO to propose an A*-IGWO for solving parking lot path planning problems. The algorithm constructs a minimum cost equation using the A* algorithm on the basis of the population updating mechanism of the IGWO, fully leveraging the advantages of both algorithms to improve performance. Dai et al. [26] proposed an improved sparrow search algorithm (ISSA) for solving the path planning problem of cellular robots on large-scale three-dimensional truss. Julius et al. [27] proposed an improved self-adaptive learning PSO (ISALPSO) algorithm for solving the path planning problem of mobile robots in 2D lidar maps. The algorithm can fine-tune the lidar information using a binary occupancy grid method during the robot's movement, thereby adaptively adjusting the parameters by ISALPSO.

The coati optimization algorithm (COA) [28] is an efficient and novel metaheuristic algorithm proposed by Dehghani et al. in 2022, widely applied to various global optimization problems. However, due to its inherent mechanism issues, it tends to fall into the dilemma of premature convergence when solving

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

high-dimensional problems. Researchers have proposed some improvement ideas for COA. Hashim et al. [29] introduced an adaptive mutation strategy for the COA to solve feature extraction and global optimization problems. Baş et al [30] proposed an Enhanced COA for solving large-scale high-dimensional big data optimization problems (BOP). Yildizdan et al. [31] used a transfer function to transform the continuous optimization problem-solving COA into a binary optimization algorithm (BinCOA) to solve the Knapsack Problem (KP) and the Uncapacitated Facility Location Problem (UFLP). Hasanien et al. [32] introduced a new improved COA to find the optimal solution for the Probability Optimal Power Flow (POPF) problem. Jia et al. [33] proposed an improved COA based on the sound search envelope strategy to solve six engineering application problems. Although DE and COA have a wide range of applications and unique advantages, both algorithms exhibit poor solution quality when dealing with high-dimensional and multimodal problems, making them susceptible to local optima. To overcome the limitations of COA, this paper combines a cross-update strategy from some DE algorithm variants with the original COA to propose a new hybrid algorithm (SDECOA). When faced with different problems, self-adaptive differential evolution-based coati optimization algorithm (SDECOA) adaptively adjusts the use frequency of update strategies, fully utilizing the update mechanisms of both algorithms to enhance the algorithm's global search and local search capabilities.

The contributions of this paper are summarized as follows:

1. Combining the update strategies from variants of DE algorithms with COA, an SDECOA is proposed, enhancing the COA's global search and local search capabilities.
2. The crossover probability CR in the DE strategy adapts dynamically based on the algorithm's iterations, increasing the convergence speed of the algorithm.
3. The proposed algorithm successfully solved 48 real-world constrained optimization problems from various fields and achieved superior results compared to the top algorithms in the CEC2020 competition.
4. The multi-robot path planning problem is NP-hard, and the presence of multiple moving obstacles further complicates the problem. The proposed algorithm effectively addresses this issue.

The organization of this paper is as follows: Section 2 discusses the relevant definitions of DE, COA, real-world constrained optimization problems, and online multi-robot path planning problems. Section 3 provides a detailed explanation of SDECOA's definition. Section 4 showcases the experimental results of this study. Section 5 is the summary and future research objectives of this paper.

## 2. Preliminaries

In this section, COA and DE algorithm are introduced separately, along with the definition of real-world constrained engineering optimization problems and the definition of multi-robot path planning problems.

### *2.1. Coati optimization algorithm (COA)*

This section describes the mathematical model of the original COA. The COA is a novel swarm intelligence algorithm proposed in 2022, which primarily simulates the behavior of coatis hunting iguanas. Initially, half of the population will climb trees to approach their food source, the iguana, and this behavior is defined by Eq. (1):

$$Xnew_i:\ xnew_{i,j} = x_{i,j} + r1 \cdot \left(Ig_j - I \cdot x_{i,j}\right), i = 1, 2, 3, \ldots \left\lfloor \frac{N}{2} \right\rfloor \tag{1}$$

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

In the above equation, $x_{i,j}$ represents the position of the $i$-th individual in the $j$-th dimension of the solution space, $Ig$ denotes the position of the best solution in the current population, $N$ represents the population size, $r_1$ is a random number between [0, 1], and $I$ takes a random value of either 1 or 2.

Faced with the siege of coatis, the iguana will randomly drop to the ground. Eq. (2) expresses this behavior in the solution space. Meanwhile, the other half of the coatis will search for their prey, as represented by Eq. (3):

$$Ig^G: Ig_j^G = lbj + r2 \cdot (ubj - lbj)\ ,\ j = 1, 2, 3, \ldots m \tag{2}$$

$$Xnew_i: xnew_{i,j} = \begin{cases} x_{i,j} + r3 \cdot \left(Ig_j^G - I \cdot x_{i,j}\right), F_{Ig}^G < F_i \\ x_{i,j} + r3 \cdot \left(I \cdot x_{i,j} - Ig_j^G\right), else \end{cases},\ i = \left\lfloor \frac{N}{2} \right\rfloor + 1, \ldots N \tag{3}$$

where $Ig_j^G$ represents the random dropping position of the iguana, with $lb$ and $ub$ denoting the upper and lower bounds of the solution space, respectively. $r_2$ and $r_3$ are both random numbers between [0,1]. In Eq. (3), $F_{Ig}^G$ represents the fitness value of $Ig$ and $F_i$ denotes the current fitness value of the $i$-th individual. Subsequently, if the fitness value $Fnew_i$ of the newly generated individual is superior to the current individual, its position is replaced; otherwise, the original position is retained, as expressed in Eq. (4):

$$X_i = \begin{cases} Xnew_i, Fnew_i < F_i \\ X_i, else \end{cases} \tag{4}$$

After locating the iguana, all the coatis will slowly surround their prey. This biological characteristic is represented by equations (5) and (6):

$$lb_j^L = \frac{lbj}{t}, ub_j^L = \frac{ubj}{t}, t = 1, 2, 3, \ldots T \tag{5}$$

$$Xnew_i: xnew_{i,j} = x_{i,j} + (1 - 2r4) \cdot \left(lb_j^L + r5 \cdot \left(ub_j^L - ub_j^L\right)\right), i = 1, 2, 3, \ldots N \tag{6}$$

where $t$ denotes the current iteration number, while $T$ represents the maximum iteration number, and $r_4$ and $r_5$ are both random numbers between [0, 1]. Subsequently, the new position of the current individual is also calculated using Eq. (4). The pseudocode of the basic framework of the COA is represented by Algorithm 1.

### *2.2. Differential evolution (DE)*

This section describes the mathematical model of original DE. DE is a well-known classic metaheuristic algorithm, mainly consisting of three evolutionary processes: mutation, crossover, and selection.

#### *2.2.1. Mutation*

In DE, the mutation process involves randomly selecting two individuals and using the difference of their position vectors as the step size for updating the target carrier position, as represented by Eq. (7):

$$y_{i,j} = x_{i,j} + F \cdot \left(x_{R1,j} - x_{R2,j}\right), i = 1, 2, 3, \ldots, N \tag{7}$$

In the above equation, $x_i$ represents the $i$-th individual in the population, $x_{R1}$ and $x_{R2}$ are two different individuals randomly selected from the population, $j$ denotes the $j$-th dimension of the solution, and $F$ is the scaling factor.

#### *2.2.2. Crossover*

The crossover process involves exchanging the $j$-th dimension component between the mutated individual $y_i$ and the target carrier $x_i$, generating a crossover individual $z_i$, as represented by Eq. (8):

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

**Algorithm 1** Pseudocode of COA.

1: Set the number of iterations $T$ and the number of coatis $N$.
2: Initialize population.
3: for $t = 1{:}T$
4: Update location of the iguana based on the location of the best member of the population.
5: Phase 1: Hunting and attacking strategy on the iguana (Exploration Phase).
6: for $i = 1 : N/2$
7: Calculate new position for the $i$-th coati using Eq. (1).
8: Update position of the $i$-th coati using Eq. (4).
9: end for
10: for $i = 1 + N/2{:}\ N$
11: Calculate random position for the iguana using Eq. (2).
12: Calculate new position for the $i$-th coati using Eq. (3).
13: Update position of the $i$-th coati using Eq. (4).
14: end for
15: Phase 2: The process of escaping from predators (Exploitation Phase).
16: Calculate the local bounds for variables.
17: for $i = 1{:}\ N$
18: Calculate the new position for the $i$-th coati using Eq. (6).
19: Update the position of the $i$-th coati using Eq. (4).
20: end for
21: Save the best candidate solution found so far.
22: end for
23: Output of the best obtained solution by COA for given problem. End COA.

$$\mathrm{Z}_i:\ z_{i,j} = \begin{cases} y_{i,j},\ r < CR | | j_r = j \\ x_{i,j},\ else \end{cases} \tag{8}$$

where $CR$ denotes the crossover probability, $j_r$ represents a randomly selected number between 1 and the maximum dimensions, and $r$ is a random number between [0, 1].

#### *2.2.3. Selection*

Similar to Eq. (4), after comparing the fitness values of the target carrier and the crossover individual, one of them is randomly selected. The better individual is chosen as the position vector for the next generation, achieving the goal of evolution:

$$X_i = \begin{cases} \mathrm{Z}_i,\ F_{z_i} < F_i \\ X_i,\ else \end{cases} \tag{9}$$

where $F_{z_i}$ represents the fitness value of the crossover individual and $F_i$ represents the fitness value of the target carrier.

### *2.3. Real-world constrained optimization problems*

This paragraph introduces the basic definition of constrained optimization problems, which typically refer to minimization problems under several equality and inequality constraints. The real-world constrained optimization problems in this paper are all derived from CEC2020, covering various fields such as industrial chemical processes, synthesis and design, mechanical engineering, power systems, power electronics, and livestock feed optimization. The definitions of these problems are referenced

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

from literature [34] and all meet the following function definition.

$$\begin{aligned} &\text{Minimize} \quad \mathrm{f}(X), X = (x_1, x_2, \ldots, x_n) \\ &\text{Subject to:} \quad \begin{aligned} &g_i(X) \le 0, i = 1, \ldots, n \\ &h_j(X) = 0, j = n+1, \ldots, m \end{aligned} \end{aligned} \tag{10}$$

In the above equation, $X$ denotes the solution vector of the problem, $g$ represents inequality constraints, $n$ denotes the number of $g$, $h$ represents equality constraints, and $m$ signifies the total number of all constraints.

### *2.4. Multi-robot online path planning problem*

This article studies the multi-robot path planning problem with multiple static and dynamic obstacles; therefore, it is necessary to consider the safe distance between robots and other robots, robots and static obstacles, and robots and dynamic obstacles. The mathematical model of this problem is provided in literature [35], where each robot has its own starting point and destination. During the robots movement, there are static and dynamic obstacles of various sizes and shapes. All robots not only need to avoid these obstacles but also must not collide with each other. The robots need to reach their destinations with the minimum cost step by step.

Minimize,

$$Fit = F_1 + F_2 + F_3 + F_4 \tag{11}$$

where $F_1$ represents the shortest distance, $F_2$ represents avoiding static obstacles, $F_3$ represents avoiding dynamic obstacles, and $F_4$ represents avoiding other robots. $F_1$ can be expressed as Eq. (12):

$$F_1 = \sum_{i=1}^{NR} (f_i + g_i) \tag{12}$$

where

$$f_i = \sqrt{(x_i^n - x_i^c)^2 + (y_i^n - y_i^c)^2} \tag{13}$$

$$g_i = \sqrt{\left(x_i^n - x_i^g\right)^2 + \left(y_i^n - y_i^g\right)^2} \tag{14}$$

where $NR$ represents the number of robots and $(x_i^g, y_i^g)$ represents the destination coordinates of the $i$-th robot. $F_2$ can be expressed as Eq. (15):

$$F_2 = \begin{cases} \varepsilon, d_i^s \le d_s \\ 0, else \end{cases} \tag{15}$$

where $\varepsilon$ is a large penalty value, $d_s$ represent the safety distance between any two objects, and $d_i^s$ is the sum of distances from all static obstacles to the $i$-th robot, as derived from Eq. (16):

$$d_i^s = \sum_{i=1}^{NR} \sum_{j=1}^{NS} \sqrt{\left(x_i^n - x_j^s\right)^2 + \left(y_i^n - y_j^s\right)^2} \tag{16}$$

where $NS$ is the number of static obstacles and $(x_j^s, y_j^s)$ is the coordinate position of the $j$-th static obstacle. $F_3$ can be expressed as Eq. (17):

$$F_3 = \begin{cases} \varepsilon, d_i^D \le d_s \\ 0, else \end{cases} \tag{17}$$

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

where $d_i^D$ is the sum of distances from all dynamic obstacles to the $i$-th robot, as obtained from Eq. (18):

$$d_i^D = \sum_{i=1}^{NR} \sum_{j=1}^{ND} \sqrt{\left(x_i^n - x_j^D\right)^2 + \left(y_i^n - y_j^D\right)^2} \tag{18}$$

where $ND$ is the number of dynamic obstacles, $(x_j^D, y_j^D)$ is the coordinate position of the $j$-th dynamic obstacle, and the movement of dynamic obstacles at each step is determined by equations (19) and (20):

$$x_j^D = x_j^D + v_j^D \cos\left(\alpha_j\right) \tag{19}$$

$$y_j^D = y_j^D + v_j^D \sin\left(\alpha_j\right) \tag{20}$$

In the above equation, $v_j^D$ represents the movement speed of dynamic obstacle $j$ and $\alpha_j$ represents its radial position relative to the target position. $F_4$ can be expressed as Eq. (21):

$$F_4 = \begin{cases} \varepsilon, d_i^R \le d_s \\ 0, else \end{cases} \tag{21}$$

where $d_i^R$ is the sum of distances between different robots and $(x_j^R, y_j^R)$ is the coordinate position of the $j$-th robot, as obtained from Eq. (22):

$$d_i^R = \sum_{i=1}^{NR-1} \sum_{j=i+1}^{NR} \sqrt{\left(x_i^n - x_j^R\right)^2 + \left(y_i^n - y_j^R\right)^2} \tag{22}$$

## 3. The proposed SDECOA algorithm

This section will elaborate on the mathematical definition of SDECOA in detail. In the original COA algorithm, there existed an imbalance between global search capability and local search capability. To overcome this deficiency, we propose called SDECOA. For optimization problems with fewer extreme points, a mutation operator with stronger local search capability should be used. For optimization problems with more extreme points, a mutation operator with stronger global and local search capability should be employed. It is difficult for a single mutation operator to balance global search capability and local search capability, creating obstacles to achieving this balance. In SDECOA, two variants of DE are introduced in the update strategy, and these strategies are called upon based on a roulette wheel method. Initially, all strategies have the same probability of being used. However, during the algorithm iteration process, strategies that yield better results are given higher usage probabilities through learning. Learning is determined through points, where a strategy that achieves good results earns a point. As a result, satisfactory results can be achieved for different problems, overcoming the challenge of imbalance between local and global search in COA.

### *3.1. Crossing*

In traditional DE algorithms, the crossing probability $CR$ is a fixed value, with literature [36] suggesting a range of values for $CR$ between [0, 1]. When $CR$ approaches zero, the changes in individuals between adjacent generations are minimal, leading to a high presence of similar individuals in the population and potentially causing the algorithm to get stuck in local optima. Conversely, when $CR$ is close to 1, the individuals update positions fluctuate greatly, hindering efficient convergence. To address these issues, this paper proposes an adaptive dynamic adjustment method, calculated as follows:

$$CR\,(\mathrm{t}) = CR_{\min} + (CR_{\max} - CR_{\min}) \cdot \left(1 - \frac{t}{T}\right) \tag{23}$$

where $CR_{\max}$ equals 0.95, $CR_{\min}$ equals 0.05, $t$ represents the current iteration number, and $T$ is the maximum number of iterations.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

### *3.2. Variant update strategies of DE*

The first introduced update strategy, *rand* < *CR*, was proposed in literature [37], while the other part was inspired by the GWO [34] and is represented by Eq. (24):

$$Xnew_i: xnew_{i,j} = \begin{cases} x_{i,j} + FF \cdot \left(x_{best,j} - x_{i,j} + x_{R1,j} - x_{R2,j}\right); if\ rand < CR \\ \frac{1}{3}\left(x_{best,j} + x_{\sec ond,j} + x_{third,j}\right), else \end{cases} \tag{24}$$

In the above equation, $x_{best}$, $x_{second}$, and $x_{third}$ represent the current population's top three individuals, *FF* is the scaling factor with a value of 0.8, while *R1* and *R2* are randomly selected individuals that are different from the *i*-th individual.

The second introduced update strategy was proposed in literature [39] and is represented by Eq. (25):

$$Xnew_i: xnew_{i,j} = x_{i,j} + L \cdot \left(x_{R1,j} - x_{i,j}\right) + FF \cdot \left(x_{R2,j} - x_{R3,j}\right) \tag{25}$$

where *L* is a random number between [0, 1], and *R1*, *R2*, and *R3* are also three distinct random individuals.

### *3.3. Algorithm implementation*

In the algorithm proposed, during the exploration phase, Eqs. (1), (24), and (25) are randomly used, with their probabilities determined by Eq. (26). During the invocation process, a counter records which update strategy an individual has been used. If an individual achieves better results in this iteration, the score of the strategy used by that individual is incremented. The probabilities for the next roulette wheel selection are calculated based on the total score obtained by each strategy in this iteration:

$$p_j(t+1) = \frac{S_j(t)}{\sum_{i=1}^{SN} S_i(t)},\ j \in \{1, 2, \ldots, SN\} \tag{26}$$

where $p_j(t+1)$ represents the probability of strategy *j* being used in the next iteration, $S_i(t)$ represents the score of strategy *i* in the current iteration, and *SN* represents the number of strategies, which is set to 3 in this paper. The update strategy for each individual is determined by a roulette wheel selection. The pseudocode and flow chart of SDECOA are, respectively, shown in Algorithm 1 and Figure 1.

### *3.4. Time complexity analysis*

The operation of SDECOA includes population initialization, fitness calculation, population position update, individual probability update, and strategy selection, so by analyzing the complexity of each process, the time complexity of SDECOA can be obtained. Assuming the population size is *N*, the number of iterations is *T*, and the dimension of the problem is *D*, the time complexity of population initialization is O(*ND*). In each iteration process, the time complexity of population position update is O(*ND*+*ND*), the time complexity of fitness value calculation and comparison is O(*ND*+*ND*), and the time complexity of individual probability update and strategy selection is O(*ND*). Therefore, the time complexity of SDECOA can be concluded as O(*ND*+*T* ×(*ND*+*ND*+*ND*+*ND*+*ND*+*ND*)) ≈ O(*TND*).

## 4. Experiment results and analysis

### *4.1. Experimental configuration and design*

The programming language used in this experiment is MATLAB, and the compilation software is MATLAB 2021a. The computer configuration consists of a 64-bit Windows 10 operating system, 8.0G RAM, and a processor with a base frequency of 2.10 GHz. To demonstrate the effectiveness of the improved algorithm, our algorithm was initially compared with several classic and advanced algorithms on the 20-dimensional CEC2022 benchmark functions. To further validate the performance of our algorithm, SDECOA was employed to solve 48 real-world constrained optimization problems from

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

**Algorithm 2** Pseudocode of SDECOA.

Set the number of iterations $T$, the number of coatis $N$ and the scale factor $FF$.
Initialize population.
$p_i \leftarrow 1/3$, $S_i \leftarrow 0$, $i \in \{1,2,3\}$.
Calculate the fitness value of the population.
for $i = 1: N$
  $pp_i \leftarrow$ RouletteWheelSelection($p$).
end for
for $t = 1:T$
  Calculate the crossover probability $CR$ using Eq. (23).
  Calculate the optimal three individuals $X_{best}$, $X_{second}$, and $X_{third}$ in the population.
  $Iguana \leftarrow X_{best}$
  for $i = 1:N$
    Randomly select three individuals R1, R2, R3 that are different from individual $i$.
    if $pp_i == 1$
      Calculate the new position $Xnew$ for the $i$-th coati using Eq. (1).
    elseif $pp_i == 2$
      Calculate the new position $Xnew$ for the $i$-th coati using Eq. (24).
    elseif $pp_i == 3$
      Calculate the new position $Xnew$ for the $i$-th coati using Eq. (25).
    end if
    Update position of the $i$-th coati using Eq. (4)
    if $f(Xnew) < f(X_i)$
      $S_j \leftarrow S_j + 1$
    end if
  end for
  Update the $p$ using Eq. (26).
  for $i = 1:N$
    $pp_i \leftarrow$ RouletteWheelSelection($p$).
  end for
  for $i = 1:N$
    Calculate the new position for the $i$-th coati using Eq. (6).
    Update the position of the $i$-th coati using Eq. (4).
  end for
    Save the best candidate solution found so far.
end for
Output of the best obtained solution by SDECOA for given problem. End SDECOA.

CEC2020, and comparisons were made with several algorithms that won in the CEC2020 competition. Finally, our algorithm was applied to the problem of multi-robot online path planning.

### *4.2. SDECOA for CEC2022 benchmark functions*

Solving benchmark test functions is a common method for testing algorithm performance, and this section presents the experimental results of SDECOA solving this test set. The CEC2022 benchmark test functions are the latest test functions, and basic information is shown in Table 1. This test suite mainly includes three dimensions ($dim = \{2, 10, 20\}$), with function types primarily including single-peak functions (F1), multimodal functions (F2–F5), hybrid functions (F6-F8), and composite functions (F9–F12). To validate the algorithm's effectiveness, we compare SDECOA with the original DE and COA.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

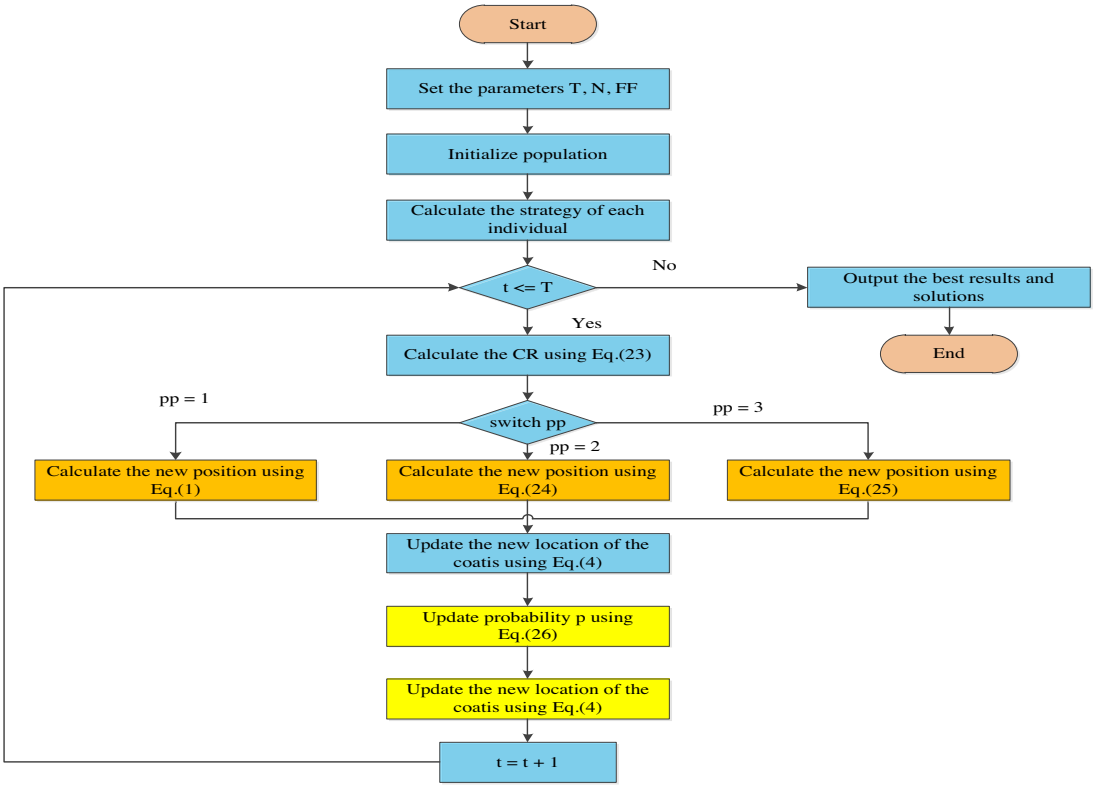


***Figure 1.*** *Flow chart of SDECOA.*

Furthermore, to evaluate its performance, we also test and analyze it against some well-known classical algorithms and recent advanced algorithms. The algorithms involved in testing the CEC2022 benchmark functions include simulated annealing (SA) [40], PSO [7], GWO [38], sine-cosine algorithm (SCA) [41], whale optimization algorithm (WOA) [42], Harris hawk optimizer (HHO) [43], SSA [44], modified adaptive sparrow search algorithm (MASSA) [41], self-adaptive differential sine-cosine algorithm (sdSCA) [35], hybrid algorithm of differential evolution and flower pollination (HADEFP) [45], etc. In this set of experiments, all algorithms were independently run 25 times. Except for SA, the iteration count for all algorithms was set to 1000 iterations, with a population size of 50. The other parameters for each algorithm are shown in Table 2.

As shown in Table 3, the means and standard deviations of different algorithms are recorded. In the experimental results across all test functions, the proposed algorithm outperforms the original COA and DE by a significant margin, particularly achieving the best results in solving F1, F6, F7, F8, and F9. Among all the compared algorithms, SDECOA proves to be the most competitive. Table 3 also presents the rankings of all algorithms based on the Friedman rank-sum test, with the data confirming that the proposed algorithm achieved the best performance in solving the CEC2022 test functions.

The convergence curves for different CEC2022 test functions are shown in Figure 2. From these curves, it is evident that SDECOA exhibits substantial improvements in both optimization capability and convergence speed compared to DE and COA. Compared with other algorithms, SDECOA shows the best convergence speed and results for most test functions. The boxplots for all test functions are shown in Figure 3, where it can be seen that SDECOA performs better in terms of median values and lower variance for most functions. Compared to other algorithms, SDECOA is more stable, thus demonstrating the superior robustness of the proposed algorithm.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table I.*** *Basic information of CEC2022 benchmark test functions.*

| | No. | Functions | $f^*$ |
|---|---|---|---|
| Unimodal Function | F1 | Shifted and full Rotated Zakharov Function | 300 |
| Basic Functions | F2 | Shifted and full Rotated Rosenbrock's Function | 400 |
| | F3 | Shifted and full Rotated Expanded Shaffer's f6 Function | 600 |
| | F4 | Shifted and full Rotated Non-continuous Rastrigin's Function | 800 |
| | F5 | Shifted and full Rotated Levy Function | 900 |
| Hybrid Functions | F6 | Hybrid Function 1 ($N = 3$) | 1800 |
| | F7 | Hybrid Function 2 ($N = 6$) | 2000 |
| | F8 | Hybrid Function 3 ($N = 5$) | 2200 |
| Composition Functions | F9 | Composition Function 1 ($N = 5$) | 2300 |
| | F10 | Composition Function 2 ($N = 4$) | 2400 |
| | F11 | Composition Function 3 ($N = 5$) | 2600 |
| | F12 | Composition Function 4 ($N = 6$) | 2700 |
| Search range: $[-100,100]^D$ | | | |

***Table II.*** *Parameter settings of different algorithms.*

| Algorithm | Parameter |
|---|---|
| SDECOA | F = 0.8 |
| DE | F = 0.4, CR = 0.5 |
| COA | – |
| SA | Initial temperature = 100, cooling rate = 0.99 |
| PSO | c1 = 2, c2 = 2, v0 = 0, w = 0.7 |
| GWO | – |
| SCA | – |
| WOA | – |
| HHO | – |
| SSA | P = 0.2, ST = 0.8, SP = 0.2 |
| MASSA | P = 0.2, ST = 0.8, SP = 0.15 |
| sdSCA | CR = 0.6, FF = 0.8 |
| HADEFP | $\alpha = 0.01$, MaxRuntime = 30, a = 4 |

### *4.3. SDECOA for CEC2020 problems*

To further demonstrate the effectiveness of SDECOA, this section presents the experimental results and data analysis of SDECOA applied to real-world constrained optimization problems. The real-world constrained problems used in this study are based on the CEC2020 real-world constrained optimization problems. The mathematical models for all problems in this study are derived from reference [46]. Table 4 displays the basic information of 48 problems, where $D$ represents the dimension of the problem, $g$ represents the number of inequality constraints, $h$ represents the number of equality constraints, and $f^*$ represents the known optimal value. These problems are mainly designed in fields such as Industrial Chemical Processes, Process Synthesis and Design, Mechanical Engineering, Power Systems, Power Electronics, and Livestock Feed Ration Optimization, with dimensions ranging from 2 to 118 and varying numbers of constraints from 1 to 108.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table III.*** *Experimental results of different algorithms on CEC2022 benchmark functions in 20 dimensions.*

| Function | Metric | SDECOA | DE | COA | SA | PSO | GWO | SCA | WOA | HHO | SSA | MASSA | sdSCA | HADEFP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 | Avg. | **3.0000E + 02** | 5.8850E + 04 | 3.7666E + 04 | 3.5105E + 04 | 1.7048E + 05 | 1.0387E + 04 | 1.3565E + 04 | 1.5429E + 04 | 2.1594E + 03 | 1.0080E + 04 | 2.1201E + 04 | 3.9211E + 02 | 1.6934E + 04 |
| | SD. | 3.4094E-03 | 3.1172E + 04 | 9.5610E + 03 | 1.5593E + 04 | 4.6289E + 04 | 4.2106E + 03 | 2.9376E + 03 | 3.9526E + 03 | 1.5431E + 03 | 5.4612E + 03 | 1.2747E + 04 | 2.3339E + 02 | 5.0041E + 03 |
| F2 | Avg. | **4.4199E + 02** | 4.8547E + 03 | 2.5294E + 03 | 4.5374E + 02 | 2.1734E + 04 | 4.8140E + 02 | 6.8423E + 02 | 5.2963E + 02 | 4.7717E + 02 | 4.4902E + 02 | 4.6164E + 02 | 4.4808E + 02 | 4.4745E + 02 |
| | SD. | 2.0764E + 01 | 1.8155E + 03 | 4.8823E + 02 | 2.1086E + 01 | 7.2428E + 03 | 2.3937E + 01 | 7.2110E + 01 | 5.0224E + 01 | 2.8911E + 01 | 9.9530E + 00 | 1.4380E + 01 | **1.**8259E+00 | 2.1303E + 00 |
| F3 | Avg. | 6.0173E + 02 | 6.9760E + 02 | 6.7513E + 02 | 6.0000E + 02 | 7.6729E + 02 | 6.0357E + 02 | 6.4424E + 02 | 6.5949E + 02 | 6.5991E + 02 | 6.5002E + 02 | 6.3845E + 02 | **6.0000E + 02** | 6.0046E + 02 |
| | SD. | 3.1131E + 00 | 1.4915E + 01 | 1.0789E + 01 | 9.6371E-03 | 1.6745E + 01 | 2.7836E + 00 | 4.8728E + 00 | 1.5219E + 01 | 9.0051E + 00 | 1.0890E + 01 | 1.6219E + 01 | 4.2874E-04 | 2.0633E-01 |
| F4 | Avg. | 8.5616E + 02 | 1.0420E + 03 | 9.6756E + 02 | 9.1241E + 02 | 1.2533E + 03 | **8.5370E + 02** | 9.3862E + 02 | 9.2024E + 02 | 8.8049E + 02 | 8.9022E + 02 | 8.8467E + 02 | 9.1549E + 02 | 9.0727E + 02 |
| | SD. | 1.2321E + 01 | 5.2455E + 01 | 1.6928E + 01 | 2.4606E + 01 | 4.8872E + 01 | 2.7739E + 01 | 1.2515E + 01 | 2.9567E + 01 | 1.2061E + 01 | 5.4475E + 00 | 1.5268E + 01 | 1.0550E + 01 | 1.6138E + 01 |
| F5 | Avg. | 1.3809E + 03 | 7.7792E + 03 | 3.2282E + 03 | 4.7036E + 03 | 2.5857E + 04 | 1.0436E + 03 | 2.1384E + 03 | 4.0273E + 03 | 2.7385E + 03 | 2.4796E + 03 | 2.2327E + 03 | **9.0000E + 02** | 9.0051E + 02 |
| | SD. | 6.1203E + 02 | 3.1415E + 03 | 3.4212E + 02 | 1.6448E + 03 | 4.1328E + 03 | 1.2666E + 02 | 2.3011E + 02 | 1.3920E + 03 | 2.7645E + 02 | 6.9158E + 01 | 4.9697E + 02 | 1.0766E-04 | 5.4120E-01 |
| F6 | Avg. | **1.8628E + 03** | 2.9397E + 09 | 1.8023E + 09 | 1.7458E + 04 | 1.2439E + 10 | 9.3085E + 05 | 1.3671E + 08 | 1.9680E + 05 | 8.1438E + 04 | 6.6033E + 03 | 5.0013E + 03 | 4.0734E + 03 | 6.9989E + 05 |
| | SD. | 2.8380E + 01 | 1.5603E + 09 | 8.4583E + 08 | 8.5441E + 03 | 6.2024E + 09 | 2.8487E + 06 | 6.9288E + 07 | 3.6333E + 05 | 3.3276E + 04 | 6.1194E + 03 | 4.1507E + 03 | 1.1220E + 04 | 7.1453E + 05 |
| F7 | Avg. | **2.0427E + 03** | 2.3026E + 03 | 2.1906E + 03 | 2.1161E + 03 | 2.4045E + 03 | 2.0750E + 03 | 2.1445E + 03 | 2.2173E + 03 | 2.1777E + 03 | 2.1671E + 03 | 2.1407E + 03 | 2.0638E + 03 | 2.0684E + 03 |
| | SD. | 1.9283E + 01 | 8.6210E + 01 | 3.7285E + 01 | 5.1414E + 01 | 4.8177E + 01 | 3.9807E + 01 | 1.9773E + 01 | 7.0364E + 01 | 5.4260E + 01 | 7.2444E + 01 | 4.3666E + 01 | 9.7407E + 00 | 1.7894E + 01 |
| F8 | Avg. | **2.2310E + 03** | 2.7497E + 03 | 2.3917E + 03 | 2.2970E + 03 | 1.6854E + 05 | 2.2540E + 03 | 2.2631E + 03 | 2.2779E + 03 | 2.2691E + 03 | 2.2955E + 03 | 2.2613E + 03 | 2.2373E + 03 | 2.2380E + 03 |
| | SD. | 2.0676E + 00 | 4.3058E + 02 | 1.1877E + 02 | 7.0938E + 01 | 1.2529E + 05 | 4.7243E + 01 | 1.4034E + 01 | 5.4650E + 01 | 5.8601E + 01 | 8.5724E + 01 | 4.5755E + 01 | 2.7204E + 00 | 2.1052E + 00 |
| F9 | Avg. | **2.4808E + 03** | 3.5370E + 03 | 3.2232E + 03 | 2.4855E + 03 | 5.2114E + 03 | 2.5091E + 03 | 2.5850E + 03 | 2.5391E + 03 | 2.4902E + 03 | 2.4809E + 03 | 2.4808E + 03 | 2.4808E + 03 | 2.4808E + 03 |
| | SD. | 6.2957E-13 | 3.2405E + 02 | 2.5265E + 02 | 4.6754E + 00 | 8.1657E + 02 | 2.1480E + 01 | 2.8873E + 01 | 3.6123E + 01 | 6.5441E + 00 | 1.5576E-01 | 2.6014E-01 | 1.1235E-08 | 1.4916E-03 |
| F10 | Avg. | 2.9324E + 03 | 5.9777E + 03 | 5.7749E + 03 | **2.4170E + 03** | 7.3046E + 03 | 2.9286E + 03 | 3.0549E + 03 | 4.5989E + 03 | 3.6398E + 03 | 3.0940E + 03 | 3.5958E + 03 | 3.3921E + 03 | 2.5260E + 03 |
| | SD. | 9.9669E + 02 | 1.5928E + 03 | 1.6254E + 03 | 2.4782E + 01 | 2.6820E + 02 | 5.9806E + 02 | 1.2903E + 03 | 1.1633E + 03 | 7.0158E + 02 | 4.4200E + 02 | 9.1648E + 02 | 7.0045E + 02 | 7.2782E + 01 |
| F11 | Avg. | 2.9040E + 03 | 6.8477E + 04 | 8.3588E + 03 | 2.9000E + 03 | 4.5224E + 05 | 3.5956E + 03 | 5.9828E + 03 | 3.1928E + 03 | 2.9867E + 03 | 2.9280E + 03 | 2.9732E + 03 | **2.9000E + 03** | 2.9543E + 03 |
| | SD. | 2.0000E + 01 | 6.2575E + 04 | 8.6341E + 02 | 3.9076E-05 | 1.2018E + 05 | 2.8798E + 02 | 4.4265E + 02 | 9.9508E + 01 | 1.0610E + 02 | 4.5823E + 01 | 1.6668E + 02 | 2.0972E-05 | 5.2048E + 01 |
| F12 | Avg. | 2.9740E + 03 | 3.7834E + 03 | 3.5750E + 03 | 2.9630E + 03 | 3.1357E + 03 | 2.9639E + 03 | 3.0482E + 03 | 3.0619E + 03 | 3.1186E + 03 | 3.0147E + 03 | 3.0857E + 03 | 2.9405E + 03 | **2.9396E + 03** |
| | SD. | 3.2828E + 01 | 3.5057E + 02 | 2.1345E + 02 | 1.7314E + 01 | 8.8622E + 01 | 1.6969E + 01 | 2.7432E + 01 | 8.6321E + 01 | 8.3877E + 01 | 6.0802E + 01 | 1.6208E + 02 | 3.5849E + 00 | 3.8063E + 00 |
| Friedman mean rank | | 2.3333 | 12.0833 | 10.8333 | 5.5833 | 12.8333 | 5.2500 | 7.7500 | 8.7500 | 7.1667 | 5.8333 | 6.0000 | 2.7500 | 3.8333 |
| Rank | | 1 | 12 | 13 | 6 | 4 | 10 | 11 | 9 | 7 | 8 | 3 | 2 | 5 |
| Statistic | | | | | | | | 107.21978 | | | | | | |

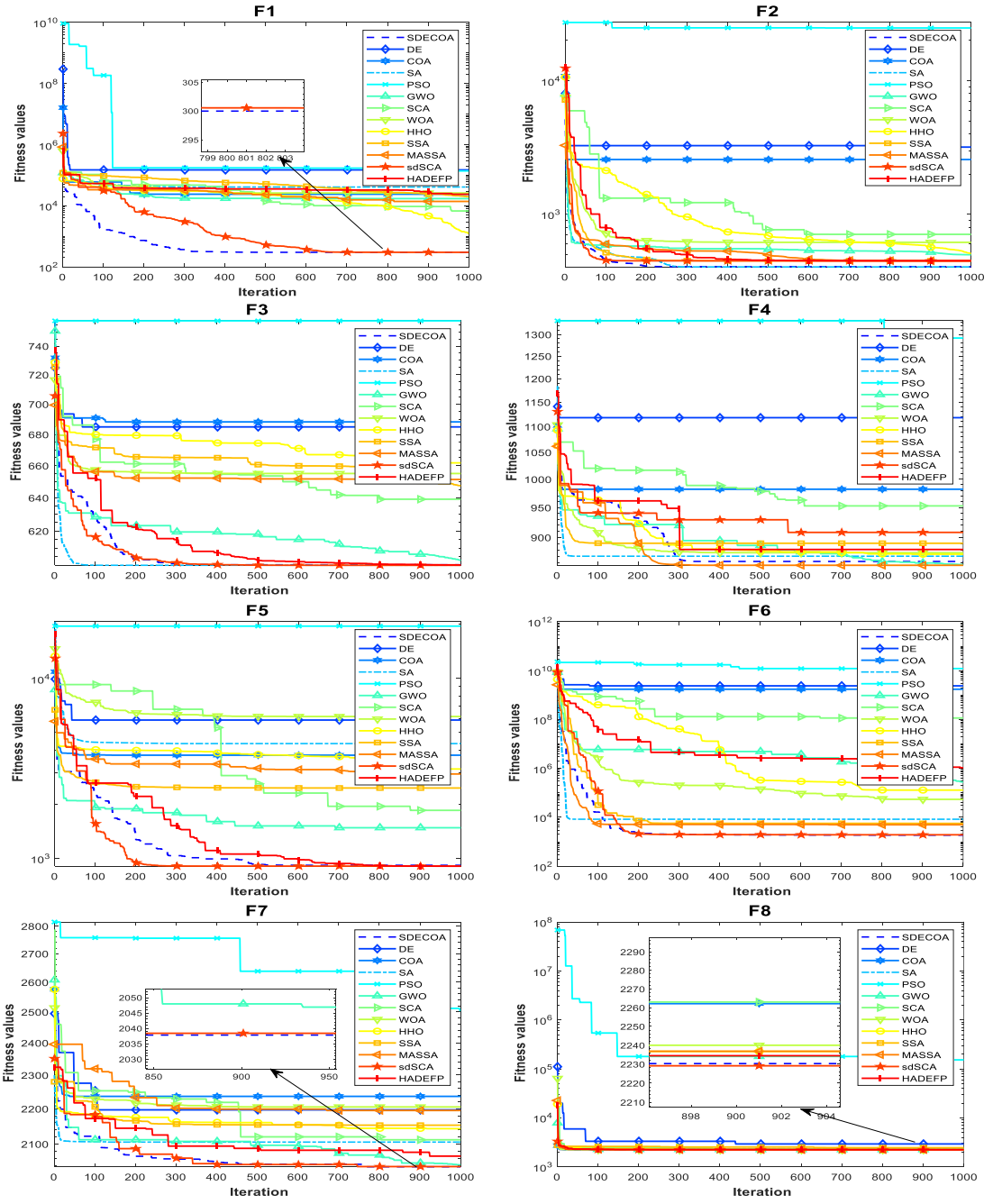


***Figure 2.*** *Convergence curves of different algorithms.*

In this set of experiments, SDECOA is compared with several advanced algorithms that won in the CEC2020 competition. These top-ranking algorithms, in descending order, include SASS [47], COLSHADE [48], sCMAgES [49], EnMODE [50], and BpMAgES [51]. Each algorithm is independently run 25 times, with 1000 iterations, a population size of 60, and other control parameters consistent with the literature [52]. A dynamic adaptive penalty function is used. Table 5 presents the optimal values, mean values, and standard deviations of these algorithms. It is worth noting that shaded cells in the

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

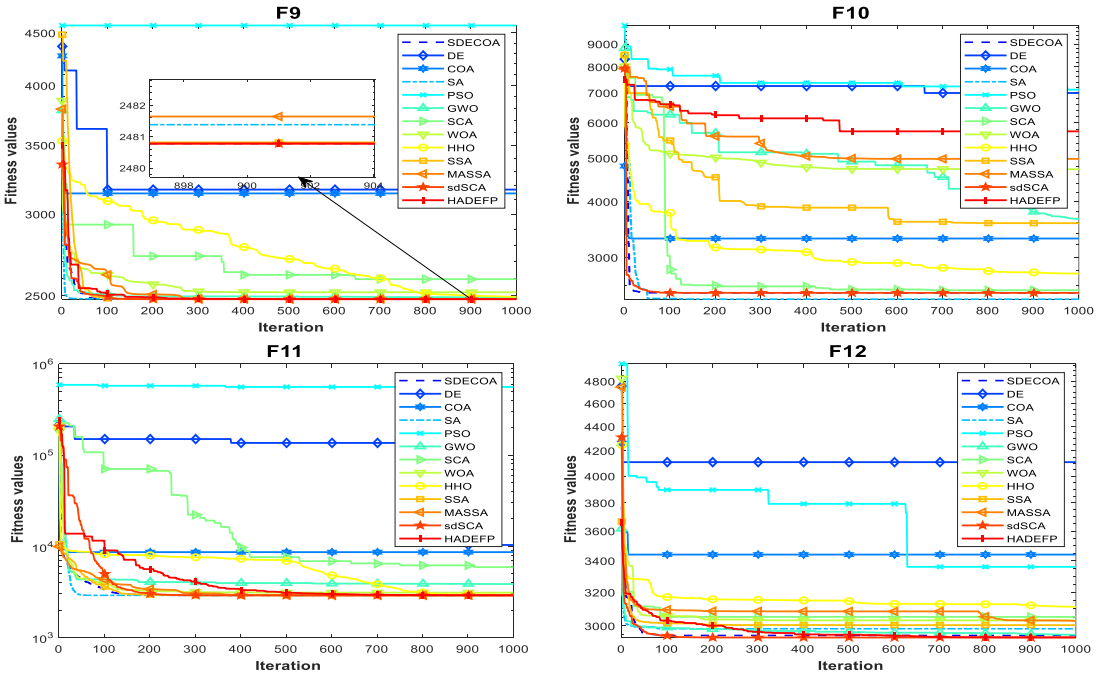


***Figure 2.*** *(Continued)*

table indicate that the proposed algorithm's best result exceeds the $f*$ for that problem. In this paper, the Friedman rank-sum test is also conducted on the average values of these 48 optimization problems, and the ranking of the proposed algorithm, obtained through calculation, is superior to the top five algorithms that won the CEC2020 competition.

The introduction in the literature [53] determines the quality of the solutions by calculating the percentage deviation of the mean with respect to the known optimal solution (*PDmean*). As can be seen from Eq. (27), the smaller the average value obtained from 25 runs, the smaller the *PDmean* value, indicating higher solution quality. Table 6 presents the *PDmean* values when different algorithms solve different problems:

$$PDmean = \begin{cases} \frac{mean - f^*}{f^* + \varepsilon}; if \quad f^* \geq 0 \\ \frac{f^* - mean}{f^*}, else \end{cases} \tag{27}$$

where $f*$ represents the best know solution, *mean* represents the average value of the test results, and $\varepsilon$ represents a very small constant. An analysis of Table 6 shows that the proposed algorithm generally demonstrates superior solution quality compared to these advanced algorithms. Based on the experimental results, the proposed algorithm achieved ideal results in most cases, with results for 26 problems surpassing the known optimal solution. These experimental results are sufficient to demonstrate that the proposed algorithm not only performs well but also excels in addressing real-world constrained optimization problems using SDECOA. Therefore, in comparison with these state-of-the-art algorithms, our algorithm demonstrates excellent competitiveness.

### *4.4. SDECOA for multi-robot path planning problem*

In this section, the experimental content applies SDECOA to the multi-robot path planning problem, the mathematical model code of which can be obtained from reference [35].

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

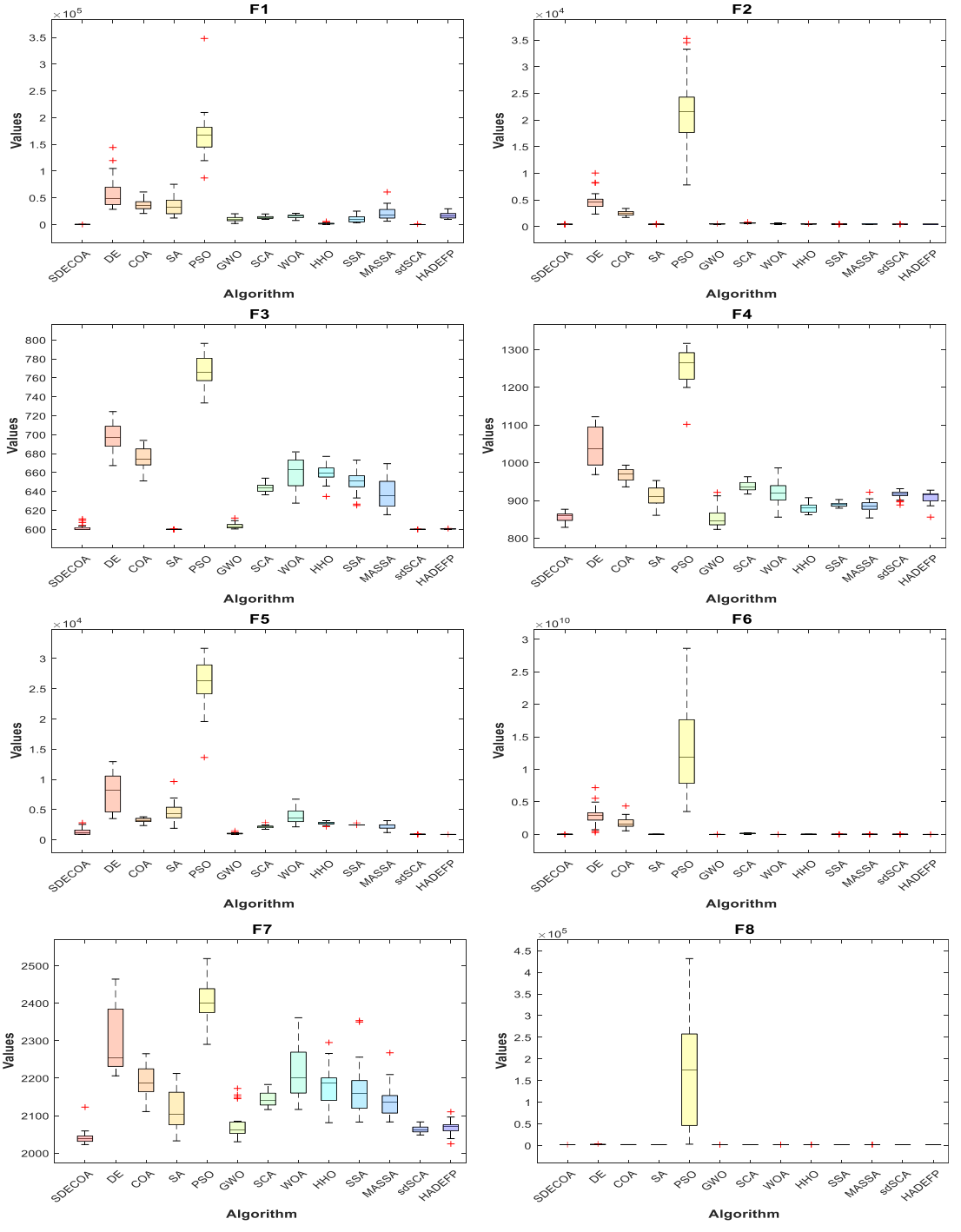


***Figure 3.*** *Boxplots of different algorithms.*

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

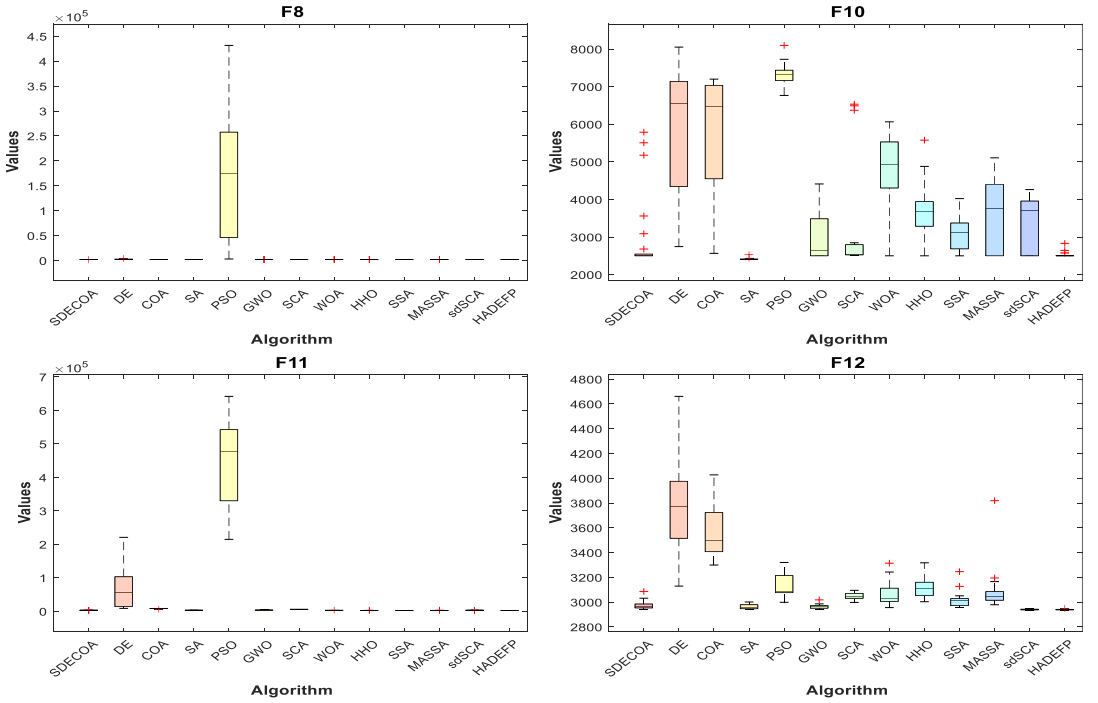


***Figure 3.*** *(Continued)*

#### *4.4.1. Design of the scenario for multi-robot path planning problem*

This section introduces the simulation scenario layout for the robot path planning problem. In the simulation model of this problem, all robots are circular with equal radii, and collisions between them are not allowed. Each robot has its own starting point and end point. As for static obstacles, each one has a different size and shape. Similarly, each dynamic obstacle is also circular with equal radii and has its own starting point and end point. Each dynamic obstacle moves at a constant speed without deviating during the motion. In this study, three different scenarios are created, with the numbers of robots, static obstacles, and dynamic obstacles shown in Table 7. Except for Scenario 1, which references literature [35], the other scenarios are arranged independently in this study. The layouts of these three scenarios are shown in Figures 4, 5, and 6. In these scenarios, the black paths represent the planned paths of robots, the blue paths represent the paths of dynamic obstacles, and '×' indicates the respective end points of objects.

In the experiments of the three scenarios mentioned above, in addition to the proposed algorithm, the comparison algorithms include DE, PSO, SCA, COA, and sdSCA, with a population size of 30, and the parameter settings are consistent with Section 4.2. It is worth noting that in this experiment, the iteration of all algorithms stops when all robots reach their end points. Each algorithm runs independently 20 times.

#### *4.4.2. Simulation experiment in Scenario 1*

In the simulation experiment in Scenario 1, Figure 7 illustrates the travel paths of each algorithm. In solving the multi-robot path planning problem, the stronger the algorithm's performance, the smoother the robot's path and the shorter the average travel distance. As seen in Figure 7, the path corresponding to SDECOA is notably the smoothest. Table 8 records the average travel distance and standard deviation for each robot from the starting point to the destination and also calculates the total average travel distance for the six robots in Scenario 1. For ease of comparison, a histogram of the average travel distances

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table IV.*** *Basic information of 48 CEC2020 real-world constrained optimization problems.*

| Problems | Name | $D$ | $g$ | $h$ | $f^*$ |
|---|---|---|---|---|---|
| | Industrial Chemical Processes | | | | |
| Pro1 | Heat Exchanger Network Design (case 1) | 9 | 0 | 8 | 1.8931163E + 02 |
| Pro2 | Heat Exchanger Network Design (case 2) | 11 | 0 | 9 | 7.0490370E + 03 |
| Pro3 | Optimal Operation of Alkylation Unit | 7 | 14 | 0 | −4.5291197E + 03 |
| Pro4 | Reactor Network Design (RND) | 6 | 1 | 4 | −3.8826044E−01 |
| Pro5 | Haverly's Pooling Problem | 9 | 2 | 4 | −4.0000560E + 02 |
| Pro6 | Blending-Pooling-Separation problem | 38 | 0 | 32 | 1.8638304E + 00 |
| Pro7 | Propane, Isobutane, n-Butane Nonsharp Separation | 48 | 0 | 38 | 2.1158628E + 00 |
| | Process Synthesis and Design Problems | | | | |
| Pro8 | Process synthesis problem | 2 | 2 | 0 | 2.0000000E + 00 |
| Pro9 | Process synthesis and design problem | 3 | 1 | 1 | 2.5576546E + 00 |
| Pro10 | Process flow sheeting problem | 3 | 3 | 0 | 1.0765431E + 00 |
| Pro11 | Two-reactor Problem | 7 | 4 | 4 | 9.9238464E + 01 |
| Pro12 | Process synthesis problem | 7 | 9 | 0 | 2.9248306E + 00 |
| Pro13 | Process design Problem | 5 | 3 | 0 | 2.6887000E + 04 |
| Pro14 | Multi-product batch plant | 10 | 10 | 0 | 5.3638943E + 04 |
| | Mechanical Engineering Problems | | | | |
| Pro15 | Weight Minimization of a Speed Reducer | 7 | 11 | 0 | 2.9944245E + 03 |
| Pro16 | Optimal Design of Industrial refrigeration System | 14 | 15 | 0 | 3.2213001E-02 |
| Pro17 | Tension/compression spring design (case 1) | 3 | 3 | 0 | 1.2665233E−02 |
| Pro18 | Pressure vessel design | 4 | 4 | 0 | 5.8853328E + 03 |
| Pro19 | Welded beam design | 4 | 5 | 0 | 1.6702177E + 00 |
| Pro20 | Three-bar truss design problem | 2 | 3 | 0 | 2.6389584E + 02 |
| Pro21 | Multiple disk clutch brake design problem | 5 | 7 | 0 | 2.3524246E−01 |
| Pro22 | Planetary gear train design optimization problem | 9 | 10 | 1 | 5.2576871E−01 |
| Pro23 | Step-cone pulley problem | 5 | 90 | 3 | 1.6069869E + 01 |
| Pro24 | Robot gripper problem | 7 | 7 | 0 | 2.5287918E + 00 |
| Pro25 | Hydro-static thrust bearing design problem | 4 | 7 | 0 | 1.6254428E + 03 |
| Pro26 | Four-stage gear box problem | 22 | 86 | 0 | 3.5359232E + 01 |
| Pro27 | 10-bar truss design | 10 | 3 | 0 | 5.2445076E + 02 |
| Pro28 | Rolling element bearing | 10 | 9 | 0 | 1.4614136E + 04 |
| Pro29 | Gas Transmission Compressor Design (GTCD) | 4 | 1 | 0 | 2.9648954E + 06 |

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table IV.*** *(Continued)*

| Problems | Name | *D* | *g* | *h* | $f^*$ |
|---|---|---|---|---|---|
| Pro30 | Tension/compression spring design (case 2) | 3 | 90 | 0 | 2.6138841E + 00 |
| Pro31 | Gear train design Problem | 4 | 1 | 1 | 0.0000000E + 00 |
| Pro32 | Himmelblau's Function | 5 | 6 | 0 | −3.0665539E + 04 |
| Pro33 | Topology Optimization | 30 | 30 | 0 | 2.6393465E + 00 |
| | Power System Problems | | | | |
| Pro34 | Optimal Sizing of Single Phase Distributed Generation with reactive power support for Phase Balancing at Main Transformer/Grid | 118 | 0 | 108 | 0.00000000E + 00 |
| Pro35 | Wind Farm Layout Problem | 30 | 91 | 0 | −6.2607000E + 03 |
| | Power Electronic Problems | | | | |
| Pro36 | SOPWM for 3-level Invereters | 25 | 24 | 1 | 3.8029251E−02 |
| Pro37 | SOPWM for 5-level Inverters | 25 | 24 | 1 | 2.1215000E−02 |
| Pro38 | SOPWM for 7-level Inverters | 25 | 24 | 1 | 1.5164538E−02 |
| Pro39 | SOPWM for 9-level Inverters | 30 | 29 | 1 | 1.6787536E−02 |
| Pro40 | SOPWM for 11-level Inverters | 30 | 29 | 1 | 9.3118742E−03 |
| Pro41 | SOPWM for 13-level Inverters | 30 | 29 | 1 | 1.5096451E−02 |
| | Livestock Feed Ration Optimization | | | | |
| Pro42 | Beef Cattle (case 1) | 59 | 14 | 1 | 4.5508511E + 03 |
| Pro43 | Beef Cattle (case 2) | 59 | 14 | 1 | 3.3489821E + 03 |
| Pro44 | Beef Cattle (case 3) | 59 | 14 | 1 | 4.9976069E + 03 |
| Pro45 | Beef Cattle (case 4) | 59 | 14 | 1 | 4.2405483E + 03 |
| Pro46 | Dairy Cattle (case 1) | 64 | 0 | 6 | 6.6964145E + 03 |
| Pro47 | Dairy Cattle (case 2) | 64 | 0 | 6 | 1.4748933E + 04 |
| Pro48 | Dairy Cattle (case 3) | 64 | 0 | 6 | 3.2132917E + 03 |

for different algorithms is shown in Figure 8. From this, it is evident that SDECOA demonstrates the best performance. Table 9 lists the number of steps taken by each robot, with fewer steps indicating a smoother path.

The experimental results from Scenario 1 show that, compared to DE and COA, the algorithm's performance improved by 33.2% and 22.4%, respectively, while the average number of steps taken was reduced by 26.9% and 15.8%.

#### *4.4.3. Simulation experiment in Scenario 2*

Figure 9 shows the travel paths of each algorithm in Scenario 2. It is clearly observed in Figure 9 that, as the number of robots increases, the differences in path smoothness between the algorithms become more pronounced. Compared to the other algorithms, the path corresponding to SDECOA is the smoothest. Table 10 records the average travel distance and corresponding standard deviation for the 12 robots from the starting point to the destination. Figure 10 presents a histogram of the average travel distances for different algorithms. From the analysis of Figure 10, it can be seen that, despite the increased complexity of the problem, SDECOA still demonstrates strong performance. Table 11 records the number of steps taken by each robot.

The experimental results from Scenario 2 show that, compared to DE and COA, the algorithm's performance improved by 46.1% and 50.2%, respectively, while the average total number of steps taken was reduced by 40.0% and 48.3%.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table V.*** *Experimental results of different algorithms solving 48 CEC2020 real-world constrained optimization problems (shaded areas indicate results superior to the known optimal solution).*

| Problems | Metric | SDECOA | SASS | COLSHADE | sCNAgES | EnMODE | BPMAgES |
|---|---|---|---|---|---|---|---|
| | Best | 1.90136494E + 02 | **1.89311639E + 02** | 1.89484062E + 02 | 1.89311647E + 02 | 1.89311726E + 02 | 1.89558919E + 02 |
| Pro1 | Avg. | 4.97261246E + 02 | **1.89311639E + 02** | 2.17274353E + 02 | 1.89311647E + 02 | 1.89311726E + 02 | 2.03797339E + 02 |
| | SD. | 2.52789483E + 02 | 5.80155714E-14 | 2.59764421E + 01 | 1.32042156E-09 | 2.90077857E-14 | 1.40425132E + 01 |
| | Best | **7.04903695E + 03** | 7.04903770E + 03 | 7.04903707E + 03 | 7.04903760E + 03 | 7.04903882E + 03 | 7.04966344E + 03 |
| Pro2 | Avg. | **7.04903695E + 03** | 7.04903770E + 03 | 7.04903707E + 03 | 7.04903760E + 03 | 7.04903882E + 03 | 1.58370588E + 04 |
| | SD. | 2.35563225E-12 | 9.28249143E-13 | 2.78474743E-12 | 2.57538931E-08 | 9.28249143E-13 | 1.06825558E + 04 |
| | Best | **−4.52918561E + 03** | −1.42719318E + 02 | −4.52911969E + 03 | −4.52776573E + 03 | −4.52911967E + 03 | −4.52911969E + 03 |
| Pro3 | Avg. | −4.34475603E + 03 | −1.42719316E + 02 | **−4.36667729E + 03** | −4.32491082E + 03 | −4.35366365E + 03 | −4.75240766E + 02 |
| | SD. | 4.27626238E + 02 | 4.08248291E-06 | 3.31584160E + 02 | 2.69220987E + 02 | 8.77280074E + 02 | 1.35106780E + 03 |
| | Best | −3.88084571E-01 | −3.88260420E-01 | **−3.88260436E-01** | −3.88244310E-01 | −3.88260375E-01 | −3.86980632E-01 |
| Pro4 | Avg. | −3.73479442E-01 | **−3.88260153E-01** | −3.87465209E-01 | −3.87155644E-01 | −3.75738899E-01 | −3.83121793E-01 |
| | SD. | 1.35723172E-02 | 5.08049538E-07 | 2.93352081E-03 | 3.70189247E-03 | 4.69053171E-03 | 3.05014349E-03 |
| | Best | −2.25000000E + 01 | −4.00005532E + 02 | −4.00005600E + 02 | −4.00003872E + 02 | **−4.00005600E + 02** | −1.96391022E + 02 |
| Pro5 | Avg. | −5.69700000E + 00 | **−4.00002118E + 02** | −3.40830432E + 02 | −3.97683750E + 02 | −3.37813699E + 02 | −3.05838116E + 01 |
| | SD. | 6.58700710E + 00 | 8.21440903E-03 | 1.17179814E + 02 | 8.00812181E + 00 | 1.31367249E + 02 | 1.38292570E + 02 |
| | Best | 1.10307109E + 00 | 1.86386710E + 00 | 1.91762880E + 00 | 1.86390003E + 00 | **1.08353014E + 00** | 2.12172047E + 00 |
| Pro6 | Avg. | 3.13774007E + 00 | 1.87233338E + 00 | 2.06339806E + 00 | 2.00520226E + 00 | **1.08701830E + 00** | 2.44867119E + 00 |
| | SD. | 2.63577263E + 00 | 1.86575140E-02 | 1.05771577E-01 | 1.32012583E-01 | 7.43075221E-03 | 1.99250965E-01 |
| | Best | 9.81574349E-01 | 1.56705180E + 00 | 1.38382662E + 00 | 1.75241050E + 00 | **9.51040119E-01** | 1.36257235E + 00 |
| Pro7 | Avg. | 2.03920690E + 00 | 1.57650206E + 00 | 1.84002526E + 00 | 2.02137918E + 00 | **1.16170831E + 00** | 1.86600951E + 00 |
| | SD. | 5.44598091E-01 | 1.98224027E-02 | 1.99058990E-01 | 1.12402096E-01 | 1.94983718E-01 | 2.96719631E-01 |
| | Best | **1.99999000E + 00** | 2.00000010E + 00 | 2.00000002E + 00 | 2.00000015E + 00 | 2.00000118E + 00 | 2.00000475E + 00 |
| Pro8 | Avg. | **1.99999000E + 00** | 2.00000010E + 00 | 2.00000002E + 00 | 2.00000015E + 00 | 2.00000118E + 00 | 2.01942277E + 00 |
| | SD. | 9.06493304E-16 | 0.00000000E + 00 | 4.53246652E-16 | 1.28197512E-15 | 1.35973996E-15 | 6.52581319E-02 |
| | Best | 2.55780959E + 00 | 2.55765484E + 00 | **2.55765461E + 00** | 2.55765478E + 00 | 2.55765487E + 00 | 2.55765815E + 00 |
| Pro9 | Avg. | 2.55780959E + 00 | 2.55765484E + 00 | **2.55765461E + 00** | 2.55765478E + 00 | 2.55765487E + 00 | 2.55766153E + 00 |
| | SD. | 1.35973996E-15 | 0.00000000E + 00 | 9.06493304E-16 | 2.77135304E-09 | 9.06493304E-16 | 7.68697762E-06 |
| | Best | **1.07652094E + 00** | 1.07654317E + 00 | 1.07654310E + 00 | 1.07654317E + 00 | 1.07654383E + 00 | 1.07655076E + 00 |
| Pro10 | Avg. | **1.07652094E + 00** | 1.07654317E + 00 | 1.10429621E + 00 | 1.07654317E + 00 | 1.15286487E + 00 | 1.07655076E + 00 |
| | SD. | 6.79869978E-16 | 2.26623326E-16 | 6.49016291E-02 | 1.94632366E-15 | 8.78772212E-02 | 2.26623326E-16 |
| | Best | 9.92395162E + 01 | **9.92384667E + 01** | 1.07781781E + 02 | 9.92384845E + 01 | 9.92385128E + 01 | 1.00285320E + 02 |

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table V.*** *(Continued)*

| Problems | Metric | SDECOA | SASS | COLSHADE | sCNAgES | EnMODE | BPMAgES |
|---|---|---|---|---|---|---|---|
| Pro11 | Avg. | 9.92441965E + 01 | 1.01516778E + 02 | 1.47815320E + 02 | **9.92388702E + 01** | 1.05096920E + 02 | 1.12439915E + 02 |
| | SD. | 3.22686049E-03 | 3.72876276E + 00 | 2.12144612E + 01 | 4.39439234E-04 | 3.72869337E + 00 | 8.99546453E + 00 |
| | Best | **2.92482984E + 00** | 2.92483078E + 00 | 2.92483061E + 00 | 2.92483069E + 00 | 2.92483153E + 00 | 2.92483930E + 00 |
| Pro12 | Avg. | 3.09538650E + 00 | 2.92483078E + 00 | **2.92483061E + 00** | 2.93621432E + 00 | 2.92483153E + 00 | 3.98163035E + 00 |
| | SD. | 4.27634626E-01 | 0.00000000E + 00 | 9.06493304E-16 | 1.12765320E-02 | 1.35973996E-15 | 7.67243223E-01 |
| | Best | **2.68680210E + 04** | 2.68874239E + 04 | 2.68874223E + 04 | 2.68874245E + 04 | 2.68874303E + 04 | 2.68874756E + 04 |
| Pro13 | Avg. | **2.68680210E + 04** | 2.68874239E + 04 | 2.68874223E + 04 | 2.68874245E + 04 | 2.68874303E + 04 | 2.68874756E + 04 |
| | SD. | 1.48519863E-11 | 7.42599314E-12 | 7.42599314E-12 | 7.42599314E-13 | 7.42599314E-12 | 3.71299657E-12 |
| | Best | **5.36349646E + 04** | 5.85054547E + 04 | 5.85054503E + 04 | 5.36511215E + 04 | 5.85054523E + 04 | 5.85562988E + 04 |
| Pro14 | Avg. | 5.81089147E + 04 | 5.85054620E + 04 | 5.85054503E + 04 | **5.66204061E + 04** | 5.85054523E + 04 | 7.59334196E + 04 |
| | SD. | 1.34654051E + 03 | 1.64760432E-02 | 0.00000000E + 00 | 3.02847983E + 03 | 1.48519863E-11 | 1.32694478E + 04 |
| | Best | **2.99423425E + 03** | 2.99442454E + 03 | 2.99442452E + 03 | 2.99442449E + 03 | 2.99442605E + 03 | 2.99443489E + 03 |
| Pro15 | Avg. | **2.99423425E + 03** | 2.99442454E + 03 | 2.99442452E + 03 | 2.99442449E + 03 | 2.99442605E + 03 | 3.00645526E + 03 |
| | SD. | 1.31274253E-13 | 9.28249143E-13 | 9.28249143E-13 | 2.50627269E-12 | **0**.00000000E + 00 | 1.54021266E + 01 |
| | Best | **3.22129956E-02** | 3.22130012E-02 | 3.22130012E-02 | 3.35503836E-02 | 3.22130314E-02 | 5.03410999E-01 |
| Pro16 | Avg. | 3.22144449E-02 | **3.22130012E-02** | 3.22130012E-02 | 3.64187643E-02 | 3.22130314E-02 | 3.86462552E + 03 |
| | SD. | 8.76687054E-07 | 7.08197893E-18 | 1.41639579E-17 | 1.76241034E-03 | 7.08197893E-18 | 1.33483803E + 04 |
| | Best | **1.26652310E-02** | 1.26652336E-02 | 1.26652330E-02 | 1.26652339E-02 | 1.26652367E-02 | 1.26653573E-02 |
| Pro17 | Avg. | **1.26652310E-02** | 1.26652336E-02 | 1.26652578E-02 | 1.26676230E-02 | 1.27104463E-02 | 1.28854549E-02 |
| | SD. | 3.99049247E-18 | 3.54098947E-18 | 1.08445455E-07 | 4.63566637E-06 | 2.01379742E-05 | 3.24045831E-04 |
| | Best | **5.74301891E + 03** | 6.05971451E + 03 | 6.05971435E + 03 | 6.05971480E + 03 | 6.05971433E + 03 | 6.05978255E + 03 |
| Pro18 | Avg. | **5.74301891E + 03** | 6.05971451E + 03 | 6.06217930E + 03 | 6.08860116E + 03 | 6.05971433E + 03 | 6.48270988E + 03 |
| | SD. | 2.78474743E-12 | 0.00000000E + 00 | 8.53142879E + 00 | 6.63653093E + 01 | 9.28249143E-13 | 4.67137882E + 02 |
| | Best | **1.67021511E + 00** | 1.67021785E + 00 | 1.67021771E + 00 | 1.67021783E + 00 | 1.67021878E + 00 | 1.67022763E + 00 |
| Pro19 | Avg. | **1.67021511E + 00** | 1.67021785E + 00 | 1.67021771E + 00 | 1.67022849E + 00 | 1.67021878E + 00 | 1.67316894E + 00 |
| | SD. | 2.07703709E-16 | 2.26623326E-16 | 4.53246652E-16 | 5.33094634E-05 | 2.26623326E-16 | 1.00550187E-02 |
| | Best | **2.63852346E + 02** | 2.63895853E + 02 | 2.63895841E + 02 | 2.63895854E + 02 | 2.63895854E + 02 | 2.63896890E + 02 |
| Pro20 | Avg. | **2.63852346E + 02** | 2.63895853E + 02 | 2.63895841E + 02 | 2.63895854E + 02 | 2.63895854E + 02 | 2.63896890E + 02 |
| | SD. | 0.00000000E + 00 | 5.80155714E-14 | 0.00000000E + 00 | 2.17058886E-12 | 1.16031143E-13 | 1.16031143E-13 |

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

*Table V. (Continued)*

| Problems | Metric | SDECOA | SASS | COLSHADE | sCNAgES | EnMODE | BPMAgES |
|---|---|---|---|---|---|---|---|
| | Best | **2.35242458E-01** | 2.35242480E-01 | 2.35242461E-01 | 2.35242468E-01 | 2.35242614E-01 | 2.35242568E-01 |
| Pro21 | Avg. | **2.35242458E-01** | 2.35242480E-01 | 2.35242461E-01 | 2.35242468E-01 | 2.35242614E-01 | 2.35242568E-01 |
| | SD. | 5.66558315E-17 | 0.00000000E + 00 | 5.66558315E-17 | 2.83279157E-17 | 5.66558315E-17 | 5.66558315E-17 |
| | Best | **5.25768707E-01** | 5.25769005E-01 | 5.25768715E-01 | 5.25967410E-01 | 5.25769222E-01 | 5.27588075E-01 |
| Pro22 | Avg. | 5.33188822E-01 | 1.11140461E + 00 | 5.41026211E-01 | 5.30809304E-01 | **5.26911917E-01** | 8.31511745E-01 |
| | SD. | 5.27604610E-03 | 8.80794458E-01 | 4.34505158E-02 | 4.34837419E-03 | 1.44018181E-03 | 4.74691245E-01 |
| | Best | **1.60439153E + 01** | 1.60698706E + 01 | 1.60698691E + 01 | 1.60699252E + 01 | 1.60698700E + 01 | 1.60700105E + 01 |
| Pro23 | Avg. | **1.60439153E + 01** | 1.60698706E + 01 | 1.60698691E + 01 | 1.62086769E + 01 | 1.60698700E + 01 | 1.60868794E + 01 |
| | SD. | 5.84670613E-15 | 7.25194643E-15 | 1.08779196E-14 | 2.05466006E-01 | 1.08779196E-14 | 2.93226549E-02 |
| | Best | **2.54378561E + 00** | 2.54378579E + 00 | 2.54378561E + 00 | 2.58013916E + 00 | 2.54378761E + 00 | 2.79518644E + 00 |
| Pro24 | Avg. | 2.54393667E + 00 | 2.54378579E + 00 | **2.54378561E + 00** | 2.82978402E + 00 | 2.54378761E + 00 | 3.82988988E + 00 |
| | SD. | 7.40084735E-04 | 1.35973996E-15 | 4.53246652E-16 | 2.20002973E-01 | 9.06493304E-16 | 1.19218086E + 00 |
| | Best | **1.24535162E + 02** | 1.61611987E + 03 | 1.61611981E + 03 | 2.28447644E + 03 | 1.61612138E + 03 | 1.69106118E + 03 |
| Pro25 | Avg. | **4.25061575E + 02** | 1.61612031E + 03 | 1.63903736E + 03 | 3.02213559E + 03 | 1.61612138E + 03 | 2.98208911E + 03 |
| | SD. | 6.07854479E + 02 | 1.14749001E-03 | 1.02805302E + 02 | 3.95554565E + 02 | 2.32062286E-13 | 1.16144570E + 03 |
| | Best | 3.62784048E + 01 | 3.62504032E + 01 | **3.53592322E + 01** | 3.62485386E + 01 | 3.53592664E + 01 | 4.20781016E + 01 |
| Pro26 | Avg. | 3.62802229E + 01 | 3.86662117E + 01 | 3.66109756E + 01 | 5.37101361E + 01 | **3.57284192E + 01** | 6.82199553E + 01 |
| | SD. | 8.81651318E-04 | 2.46204679E + 00 | 1.39591172E + 00 | 1.78764043E + 01 | 5.99105585E-01 | 1.78825339E + 01 |
| | Best | **5.22383263E + 02** | 5.24457638E + 02 | 5.24450764E + 02 | 5.24508224E + 02 | 5.24451210E + 02 | 5.24471114E + 02 |
| Pro27 | Avg. | **5.22383263E + 02** | 5.24469909E + 02 | 5.24450764E + 02 | 5.24740013E + 02 | 5.24451210E + 02 | 5.28368863E + 02 |
| | SD. | 3.00890125E-07 | 6.99795427E-03 | 2.32062286E-13 | 1.91943684E-01 | 0.00000000E + 00 | 2.92877989E + 00 |
| | Best | **1.46078146E + 04** | 1.46141364E + 04 | 1.69582022E + 04 | 1.46141363E + 04 | 1.69582084E + 04 | 1.46241749E + 04 |
| Pro28 | Avg. | **1.46078146E + 04** | 1.46141364E + 04 | 1.69582022E + 04 | 1.46141363E + 04 | 1.69582084E + 04 | 1.50840552E + 04 |
| | SD. | 4.06738396E-12 | 3.71299657E-12 | 0.00000000E + 00 | 1.35257081E-11 | 7.42599314E-12 | 9.69919837E + 02 |
| | Best | **1.67775928E + 06** | 2.96489564E + 06 | 2.96489542E + 06 | 2.96489558E + 06 | 2.96489619E + 06 | 2.96490195E + 06 |
| Pro29 | Avg. | **1.67807821E + 06** | 2.96489564E + 06 | 2.96489542E + 06 | 2.96491252E + 06 | 2.96489619E + 06 | 2.96507075E + 06 |
| | SD. | 1.59468037E + 03 | 1.42579068E-09 | 4.75263561E-10 | 3.48061660E + 01 | 1.42579068E-09 | 5.80107481E + 02 |
| | Best | 2.65855791E + 00 | 2.65855930E + 00 | 2.65855921E + 00 | 2.85234342E + 00 | 2.65856093E + 00 | **2.61390821E + 00** |
| Pro30 | Avg. | 2.65855791E + 00 | 2.65855930E + 00 | 2.66183397E + 00 | 4.23690707E + 00 | 2.81494085E + 00 | **2.62819070E + 00** |
| | SD. | 1.35973996E-15 | 4.53246652E-16 | 1.13342498E-02 | 1.05662763E + 00 | 3.65700052E-01 | 3.30602014E-02 |

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table V.*** *(Continued)*

| Problems | Metric | SDECOA | SASS | COLSHADE | sCNAgES | EnMODE | BPMAgES |
|---|---|---|---|---|---|---|---|
| | Best | **0.00000000E + 00** | 2.88828925E-24 | 1.15940219E-23 | **0.00000000E + 00** | **0.00000000E + 00** | **0.00000000E + 00** |
| Pro31 | Avg. | 6.07355164E-16 | 3.60715445E-18 | 1.88066664E-16 | **0.00000000E + 00** | **0.00000000E + 00** | **0.00000000E + 00** |
| | SD. | 9.86793323E-16 | 1.24318244E-17 | 3.89233887E-16 | 0.00000000E + 00 | 0.00000000E + 00 | 0.00000000E + 00 |
| | Best | **−3.06675816E + 04** | −3.06655384E + 04 | −3.06655388E + 04 | −3.06655365E + 04 | −3.06655389E + 04 | −3.06653717E + 04 |
| Pro32 | Avg. | **−3.06675816E + 04** | −3.06655384E + 04 | −3.06655388E + 04 | −3.06655365E + 04 | −3.06655389E + 04 | −3.06652139E + 04 |
| | SD. | 7.38876989E-12 | 1.11389897E-11 | 1.11389897E-11 | 3.14181826E-11 | 7.42599314E-12 | 6.84538805E-01 |
| | Best | **2.63934650E + 00** | 2.63934651E + 00 | 2.63934651E + 00 | 2.63934659E + 00 | 2.63934847E + 00 | 2.63934979E + 00 |
| Pro33 | Avg. | **2.63934650E + 00** | 2.63934651E + 00 | 2.63934651E + 00 | 2.63934659E + 00 | 2.63934847E + 00 | 2.71534725E + 00 |
| | SD. | 9.06493304E-16 | 0.00000000E + 00 | 0.00000000E + 00 | 1.10279670E-15 | 9.06493304E-16 | 4.95061367E-02 |
| | Best | 2.26165480E-01 | **9.88911680E-10** | 1.24281937E + 00 | 3.26365242E-01 | 5.85369556E-06 | 4.66777204E-01 |
| Pro34 | Avg. | 6.68733178E-01 | **1.18911605E-03** | 4.95481969E + 00 | 1.22930749E + 00 | 2.93235236E + 00 | 9.19360334E-01 |
| | SD. | 2.71788762E-01 | 3.37986831E-03 | 2.05741650E + 00 | 5.73900722E-01 | 3.50850049E + 00 | 3.20170970E-01 |
| | Best | −5.91780159E + 03 | −6.21107481E + 03 | −6.19728069E + 03 | **−6.22250950E + 03** | −6.18684896E + 03 | −5.94718427E + 03 |
| Pro35 | Avg. | −5.48530985E + 03 | **−6.09804308E + 03** | −6.03241907E + 03 | −6.08570456E + 03 | −6.08384270E + 03 | −5.61595587E + 03 |
| | SD. | 1.40015782E + 02 | 8.72131893E + 01 | 1.08443435E + 02 | 8.01671935E + 01 | 5.34931484E + 01 | 3.41836051E + 02 |
| | Best | 2.38528540E-01 | **3.14100567E-02** | 3.43680404E-02 | 6.22341065E-02 | 7.43337014E-02 | 7.36185387E-02 |
| Pro36 | Avg. | 4.88500237E-01 | 5.26138823E-02 | **4.27949467E-02** | 8.56817526E-02 | 1.43239307E-01 | 8.52844563E-01 |
| | SD. | 1.94719298E-01 | 1.02129100E-02 | 5.63196460E-03 | 1.68405113E-02 | 5.43043874E-02 | 4.56265888E-01 |
| | Best | 1.06153620E-01 | 3.73003372E-02 | **2.02403353E-02** | 2.70744540E-02 | 5.15082868E-02 | 1.89802144E-01 |
| Pro37 | Avg. | 2.37090658E-01 | 5.55104749E-02 | **2.60819335E-02** | 4.87614434E-02 | 6.35807994E-02 | 4.46911854E-01 |
| | SD. | 5.85386432E-02 | 1.14318075E-02 | 5.79570143E-03 | 1.55700510E-02 | 5.17854199E-03 | 2.91587900E-01 |
| | Best | 1.03678745E-01 | 2.10371838E-02 | **1.27830682E-02** | 2.18177361E-02 | 3.81537441E-02 | 4.58417421E-02 |
| Pro38 | Avg. | 1.03682993E-01 | 4.90818129E-02 | **1.82115863E-02** | 3.32366546E-02 | 6.43661599E-02 | 4.77230818E-01 |
| | SD. | 2.40870981E-06 | 2.96663478E-02 | 3.26107960E-03 | 6.12417268E-03 | 1.68774575E-02 | 4.31820157E-01 |
| | Best | 3.01636422E-01 | 2.16197578E-02 | **1.68273483E-02** | 2.50975344E-02 | 4.03635428E-02 | 2.64446615E-01 |
| Pro39 | Avg. | 7.33351813E-01 | 5.99707655E-02 | **2.18758966E-02** | 1.33218921E-01 | 7.00629241E-02 | 5.58620218E-01 |
| | SD. | 2.41776910E-01 | 2.26586180E-02 | 4.07834543E-03 | 1.96907336E-01 | 1.00227369E-01 | 3.43224331E-01 |
| | Best | 1.06883388E-01 | 2.29202003E-02 | **2.17177091E-02** | 4.39293017E-02 | 3.73134549E-02 | 1.15740197E-01 |
| Pro40 | Avg. | 2.98612456E-01 | 3.77571453E-02 | **3.25815543E-02** | 1.80061930E-01 | 9.41501471E-02 | 3.58325090E-01 |
| | SD. | 1.00324480E-01 | 9.86770922E-03 | 4.15778833E-03 | 9.00013900E-02 | 4.80954246E-02 | 1.95955310E-01 |

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table V.*** *(Continued)*

| Problems | Metric | SDECOA | SASS | COLSHADE | sCNAgES | EnMODE | BPMAgES |
|---|---|---|---|---|---|---|---|
| | Best | 1.52823432E-01 | **1.51151681E-02** | 2.04502232E-02 | 1.79851064E-02 | 1.36541786E-01 | 8.19117274E-02 |
| Pro41 | Avg. | 3.38185299E-01 | **2.26010046E-02** | 6.50911405E-02 | 8.38493866E-02 | 3.27215685E-01 | 2.09060523E-01 |
| | SD. | 6.19263210E-02 | 7.12209847E-03 | 4.91924745E-02 | 2.69146197E-02 | 7.07874248E-02 | 7.94531560E-02 |
| | Best | 4.50901256E + 03 | 4.55091819E + 03 | 4.55089302E + 03 | 4.55085207E + 03 | 4.45822736E + 03 | **3.65750914E + 03** |
| Pro42 | Avg. | 4.50922981E + 03 | 4.55098126E + 03 | 4.55094514E + 03 | 4.55100407E + 03 | **4.50300277E + 03** | 5.80365461E + 03 |
| | SD. | 1.33101751E-01 | 7.16906300E-02 | 6.92445105E-02 | 1.04759010E-01 | 1.81981927E + 01 | 2.26501373E + 03 |
| | Best | 3.58129428E + 03 | 3.75088403E + 03 | 3.35225321E + 03 | 3.58929248E + 03 | **3.34971895E + 03** | 4.85718147E + 03 |
| Pro43 | Avg. | 4.26834374E + 03 | 4.19099441E + 03 | 3.37212476E + 03 | 3.63393847E + 03 | **3.36815435E + 03** | 5.92391310E + 03 |
| | SD. | 4.54325201E + 02 | 2.81548423E + 02 | 1.32470824E + 01 | 5.60554017E + 01 | 1.51924733E + 01 | 1.14774009E + 03 |
| | Best | 5.18652247E + 03 | 5.00123129E + 03 | 5.02952031E + 03 | 5.15773450E + 03 | **4.34524830E + 03** | 4.43794820E + 03 |
| Pro44 | Avg. | 5.18678939E + 03 | 5.25637355E + 03 | 5.10949966E + 03 | 5.46661366E + 03 | **4.67610664E + 03** | 5.67576975E + 03 |
| | SD. | 1.63985769E-01 | 1.54961347E + 02 | 5.76759871E + 01 | 1.15641334E + 02 | 4.31813269E + 02 | 9.80924315E + 02 |
| | Best | 3.59624113E + 03 | 4.24054431E + 03 | 4.24166502E + 03 | 4.24608539E + 03 | 3.19205843E + 03 | **2.46363995E + 03** |
| Pro45 | Avg. | 3.59640071E + 03 | 4.24153092E + 03 | 4.24593645E + 03 | 4.27310050E + 03 | **3.33474725E + 03** | 5.52347602E + 03 |
| | SD. | 1.09825017E-01 | 2.97891024E + 00 | 3.47940766E + 00 | 8.56411494E + 00 | 3.00734113E + 01 | 3.35382785E + 03 |
| | Best | 3.48444483E + 03 | 6.69707089E + 03 | 6.69800955E + 03 | 6.72712930E + 03 | **1.69427926E + 03** | 3.53979682E + 03 |
| Pro46 | Avg. | 4.96537944E + 03 | 6.70061243E + 03 | 6.73250538E + 03 | 6.72737549E + 03 | **4.93755208E + 03** | 6.09141713E + 03 |
| | SD. | 1.00978715E + 03 | 2.52978893E + 00 | 5.57923062E + 01 | 2.83448059E-01 | 1.68757424E + 03 | 1.89471056E + 03 |
| | Best | 1.03918716E + 04 | 1.47465804E + 04 | 1.40125061E + 04 | 1.47534654E + 04 | **9.04906878E + 03** | 1.08239247E + 04 |
| Pro47 | Avg. | 1.36187779E + 04 | 1.47518351E + 04 | 1.46466559E + 04 | 1.47647685E + 04 | **1.14192669E + 04** | 1.35606103E + 04 |
| | SD. | 1.74646343E + 03 | 3.92778171E + 00 | 2.08954188E + 02 | 3.30313032E + 00 | 1.21731025E + 03 | 1.87793072E + 03 |
| | Best | 2.57867513E + 03 | 3.21329173E + 03 | 3.29508063E + 03 | 3.24899607E + 03 | **1.96266818E + 03** | 3.29133159E + 03 |
| Pro48 | Avg. | 5.19002448E + 03 | 3.21330864E + 03 | 3.62823993E + 03 | 3.31230400E + 03 | **2.46861402E + 03** | 7.46744502E + 03 |
| | SD. | 2.14966966E + 03 | 3.99869908E-02 | 2.99257683E + 02 | 4.51513658E + 01 | 3.74142744E + 02 | 2.35293584E + 03 |
| Friedman mean rank | | 2.6771 | 3.2917 | 2.8542 | 3.8646 | 3.4479 | 4.8646 |
| Rank | | 1 | 3 | 2 | 5 | 4 | 6 |

***Table VI.*** *Percentage deviation values (PDmean) obtained through solutions using different algorithms.*

| Problems | SDECOA | SASS | COLSHADE | sCNAgES | EnMODE | BPMAgES |
|---|---|---|---|---|---|---|
| Pro1 | 162.668 | 0.000 | 14.771 | 0.000 | 0.000 | 7.652 |
| Pro2 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 124.670 |
| Pro3 | 4.071 | 96.849 | 3.587 | 4.509 | 3.874 | 110.493 |
| Pro4 | 3.807 | 0.000 | 0.205 | 0.285 | 3.225 | 1.324 |
| Pro5 | 98.576 | 0.001 | 14.794 | 0.580 | 15.548 | 92.354 |
| Pro6 | 68.349 | 0.456 | 10.707 | 7.585 | −41.678 | 31.378 |
| Pro7 | −3.623 | −25.491 | −13.037 | −4.465 | −45.095 | −11.809 |
| Pro8 | −0.001 | 0.000 | 0.000 | 0.000 | 0.000 | 0.971 |
| Pro9 | 0.006 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Pro10 | −0.002 | 0.000 | 2.578 | 0.000 | 7.090 | 0.001 |
| Pro11 | 0.006 | 2.296 | 48.950 | 0.000 | 5.903 | 13.303 |
| Pro12 | 5.831 | 0.000 | 0.000 | 0.389 | 0.000 | 36.132 |
| Pro13 | −0.071 | 0.002 | 0.002 | 0.002 | 0.002 | 0.002 |
| Pro14 | 8.333 | 9.073 | 9.073 | 5.558 | 9.073 | 41.564 |
| Pro15 | −0.006 | 0.000 | 0.000 | 0.000 | 0.000 | 0.402 |
| Pro16 | 0.004 | 0.000 | 0.000 | 13.056 | 0.000 | 0.000 |
| Pro17 | 0.000 | 0.000 | 0.000 | 0.019 | 0.357 | 1.739 |
| Pro18 | −2.418 | 2.963 | 3.005 | 3.454 | 2.963 | 10.150 |
| Pro19 | 0.000 | 0.000 | 0.000 | 0.001 | 0.000 | 0.177 |
| Pro20 | −0.016 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Pro21 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Pro22 | 1.411 | 111.387 | 2.902 | 0.959 | 0.217 | 58.152 |
| Pro23 | −0.162 | 0.000 | 0.000 | 0.864 | 0.000 | 0.106 |
| Pro24 | 0.599 | 0.593 | 0.593 | 11.903 | 0.593 | 51.451 |
| Pro25 | −73.849 | −0.574 | 0.836 | 85.927 | −0.573 | 83.463 |
| Pro26 | 2.605 | 9.353 | 3.540 | 51.898 | 1.044 | 92.934 |
| Pro27 | −0.394 | 0.004 | 0.000 | 0.055 | 0.000 | 0.747 |
| Pro28 | −0.043 | 0.000 | 16.040 | 0.000 | 16.040 | 3.216 |
| Pro29 | −43.402 | 0.000 | 0.000 | 0.001 | 0.000 | 0.006 |
| Pro30 | 1.709 | 1.709 | 1.834 | 62.092 | 7.692 | 0.547 |
| Pro31 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Pro32 | −0.007 | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 |
| Pro33 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 2.880 |
| Pro34 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Pro35 | 12.385 | 2.598 | 3.646 | 2.795 | 2.825 | 10.298 |
| Pro36 | 1184.538 | 38.351 | 12.532 | 125.305 | 276.656 | 2142.602 |
| Pro37 | 1017.561 | 161.657 | 22.941 | 129.844 | 199.697 | 2006.584 |
| Pro38 | 1907.595 | 850.362 | 252.628 | 543.555 | 1146.310 | 9140.532 |
| Pro39 | 4268.430 | 257.234 | 30.310 | 693.559 | 317.351 | 3227.589 |
| Pro40 | 3106.792 | 305.473 | 249.893 | 1833.681 | 911.076 | 3748.045 |
| Pro41 | 2140.164 | 49.711 | 331.168 | 455.424 | 2067.501 | 1284.832 |
| Pro42 | −0.915 | 0.003 | 0.002 | 0.003 | −1.051 | 27.529 |
| Pro43 | 27.452 | 25.142 | 0.691 | 8.509 | 0.572 | 76.887 |
| Pro44 | 3.785 | 5.178 | 2.239 | 9.385 | −6.433 | 13.570 |

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table VI.*** *(Continued)*

| Problems | SDECOA | SASS | COLSHADE | sCNAgES | EnMODE | BPMAgES |
|---|---|---|---|---|---|---|
| Pro45 | −15.190 | 0.023 | 0.127 | 0.768 | −21.360 | 30.254 |
| Pro46 | −25.850 | 0.063 | 0.539 | 0.462 | −26.266 | −9.035 |
| Pro47 | −7.663 | 0.020 | −0.693 | 0.107 | −22.576 | −8.057 |
| Pro48 | 61.517 | 0.001 | 12.913 | 3.081 | −23.175 | 132.392 |

***Table VII.*** *Number of robots and obstacles in different scenarios.*

| Scenario | Robots | Static obstacles | Dynamic obstacles |
|---|---|---|---|
| Scenario 1 | 6 | 7 | 3 |
| Scenario 2 | 12 | 9 | 5 |
| Scenario 3 | 20 | 13 | 7 |

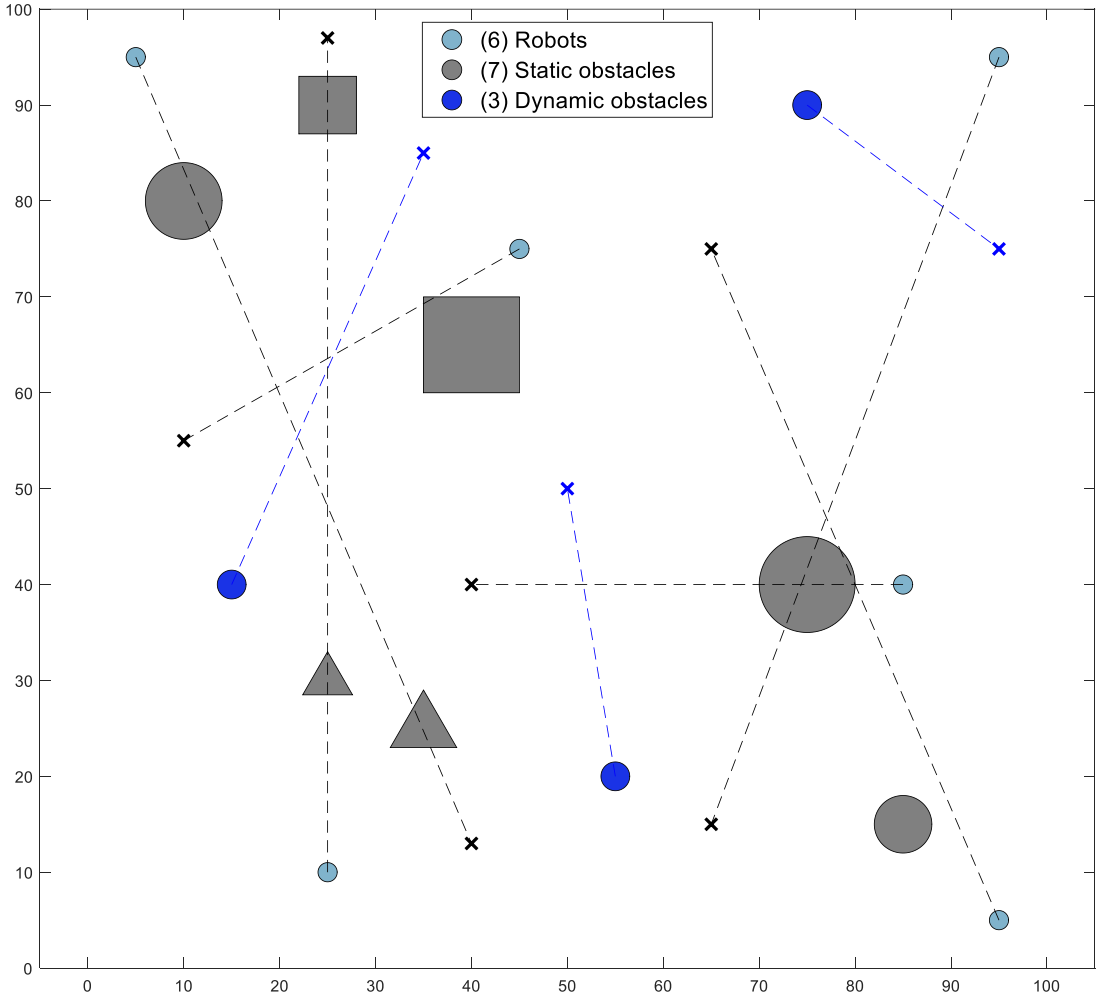


***Figure 4.*** *Scenario 1.*

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

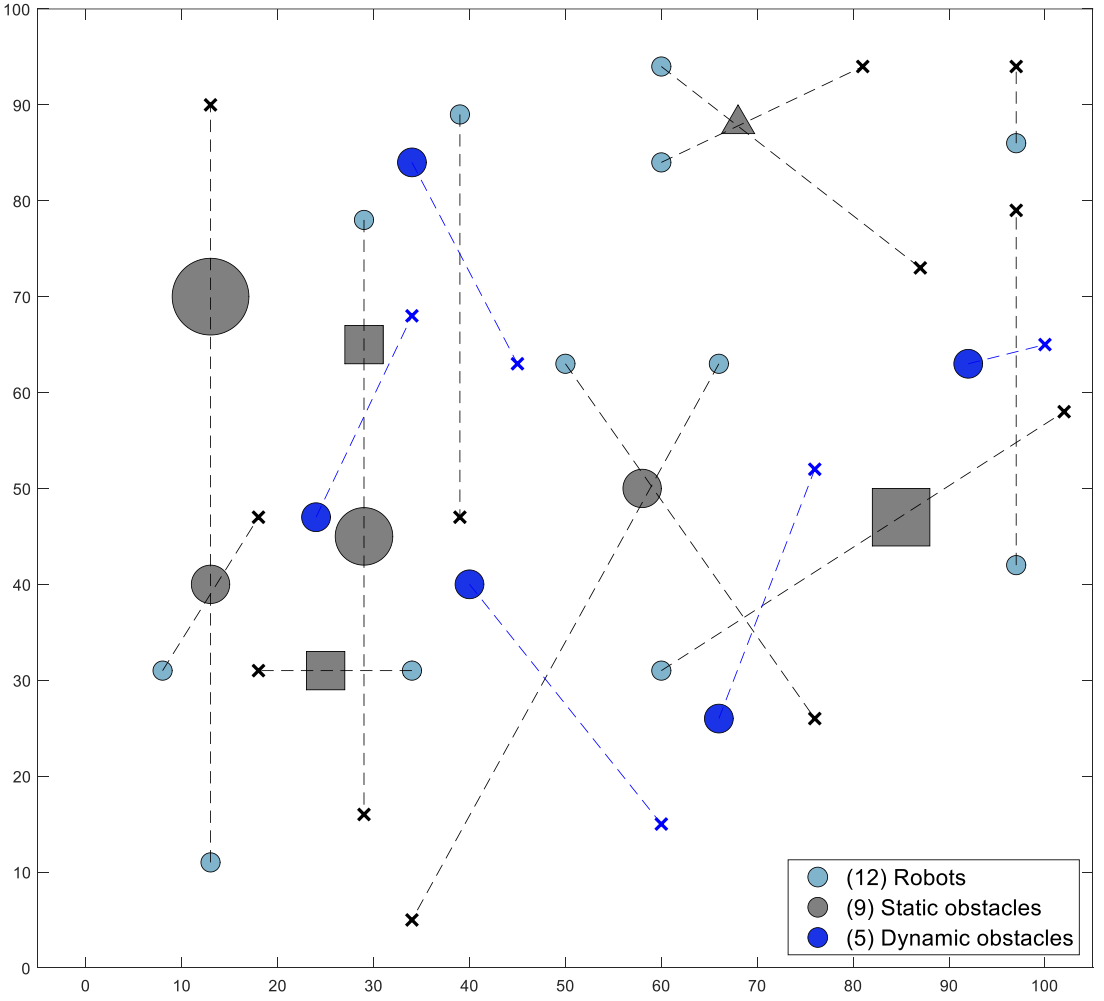


***Figure 5.*** *Scenario 2.*

#### *4.4.4. Simulation experiment in Scenario 3*

In Scenario 3, the number of robots was increased to 20, and the quantities of both dynamic and static obstacles were significantly raised, making the path planning problem more complex. It should be noted that in this experiment, COA led to the phenomenon of robots getting lost, demonstrating subpar performance. Therefore, the experimental results involving COA are excluded from consideration in this section. In this experiment, Figure 11 illustrates the trajectories of each algorithm in Scenario 3. From Figure 11, it is clearly observable that, even as the problem scale increased, the proposed algorithm still managed to find effective paths compared to the other algorithms. Table 12 records the average travel distance and corresponding standard deviation for 20 robots from the starting point to the destination, while Figure 12 shows the histogram of the average travel distances for different algorithms. The histogram clearly indicates that the average travel distance of SDECOA is significantly lower than that of the other algorithms. Table 13 documents the number of steps taken by each robot.

From the experimental results in Scenario 3, it can be concluded that, compared to DE, the proposed algorithm improved performance by 45.3%, with the total average movement steps reduced by 37.0%.

Through these experimental data, it can be seen that regardless of the number of robots, the proposed algorithm always achieves the best performance. Moreover, as the number of robots increases, the values

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

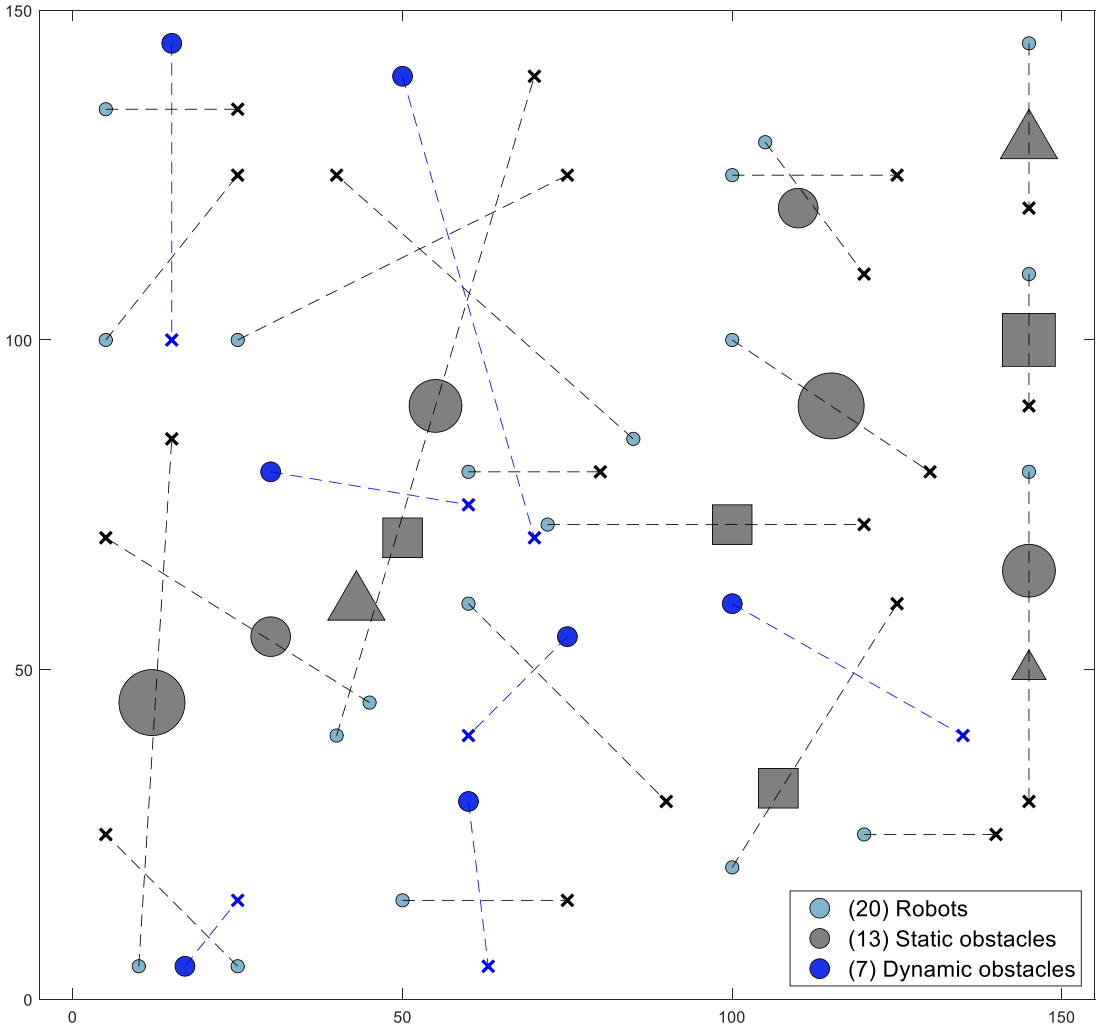


***Figure 6.*** *Scenario 3.*

of other algorithms become very large, while SDECOA can still manage reasonably. The path diagrams in the three scenarios clearly show that the proposed algorithm's results in a shorter travel distance and smoother paths for the robot, particularly demonstrating significant advantages when the robot operates in more complex environments. Therefore, it can be proven that our algorithm outperforms these algorithms in terms of performance and competitiveness.

## 5. Conclusion and future work

Current research on robot path planning mainly considers environments with static obstacles or involves a very small number of robots. However, in this study, we consider a scenario with 20 robots and 7 dynamic obstacles, significantly increasing the complexity of the problem. To address the issue of multi-robot online path planning in environments with both static and dynamic obstacles, this paper proposed a SDECOA. The proposed algorithm extends the original COA by incorporating two DE update strategies. These strategies are adaptively selected based on a learning mechanism, which eliminates the

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

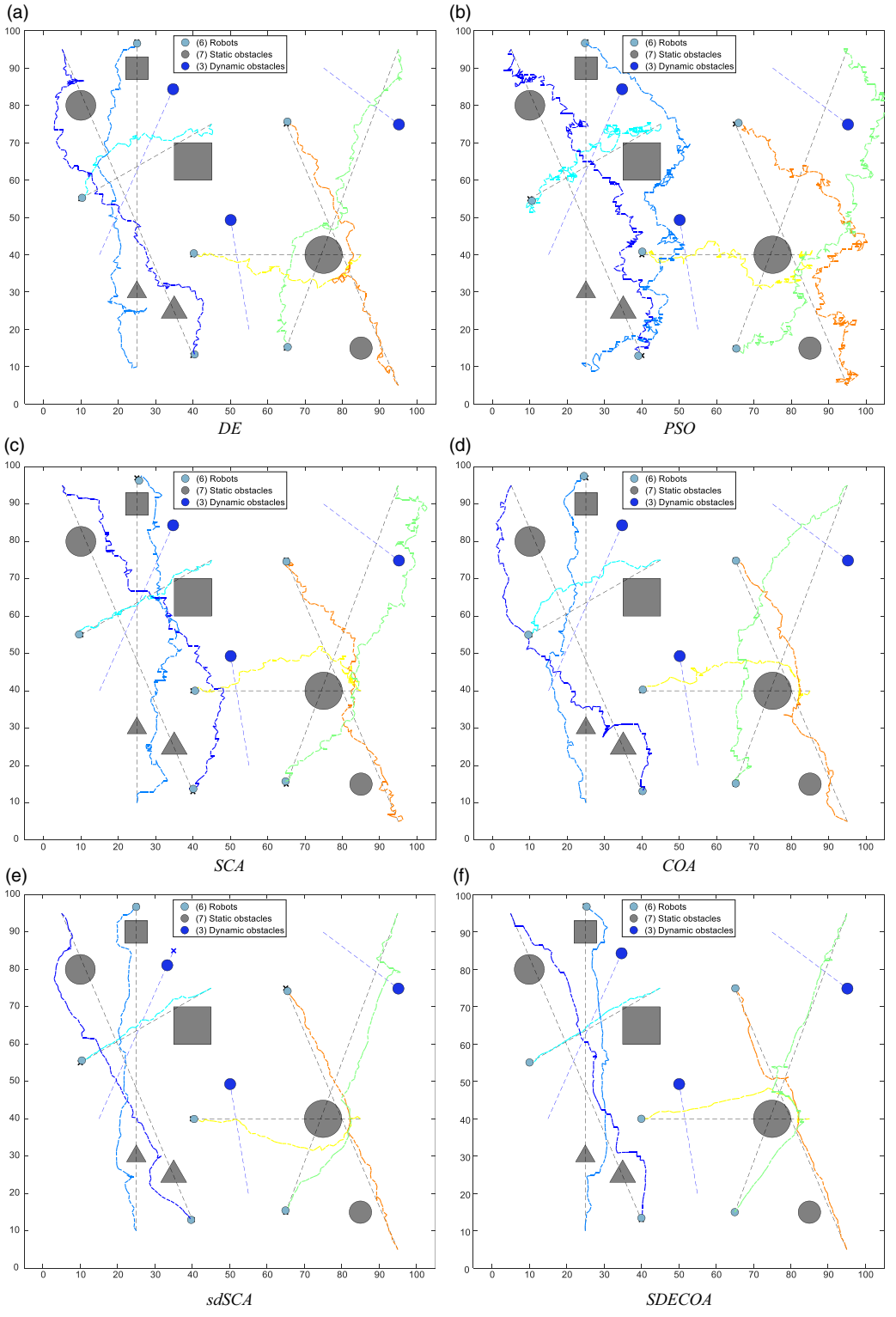


***Figure 7.*** *Paths of different algorithms in Scenario 1.*

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table VIII.*** *Experimental results of the travel distance for each robot in Scenario 1.*

| Robot | Metric | DE | PSO | SCA | COA | sdSCA | SDECOA |
|---|---|---|---|---|---|---|---|
| R1 | Avg. | 155.298 | 312.732 | 158.077 | 145.165 | 109.556 | 100.902 |
| | SD. | 7.996 | 22.151 | 3.643 | 10.923 | 3.996 | 0.765 |
| R2 | Avg. | 142.451 | 371.126 | 177.660 | 116.102 | 105.660 | 98.892 |
| | SD. | 6.006 | 27.587 | 7.871 | 5.198 | 3.487 | 5.980 |
| R3 | Avg. | 64.119 | 250.800 | 85.025 | 53.376 | 44.706 | 39.850 |
| | SD. | 8.620 | 48.552 | 15.284 | 2.519 | 1.199 | 0.590 |
| R4 | Avg. | 143.411 | 339.231 | 162.100 | 120.651 | 104.692 | 100.486 |
| | SD. | 6.713 | 32.274 | 8.103 | 8.223 | 1.664 | 5.814 |
| R5 | Avg. | 84.838 | 249.738 | 92.081 | 71.090 | 55.934 | 56.952 |
| | SD. | 5.273 | 37.285 | 10.466 | 7.797 | 1.720 | 5.786 |
| R6 | Avg. | 119.637 | 311.692 | 144.330 | 104.418 | 82.244 | 77.066 |
| | SD. | 5.506 | 16.923 | 9.404 | 7.127 | 1.883 | 1.660 |
| Total mean distance | 709.755 | 1835.320 | 819.274 | 610.801 | 502.792 | 474.149 | |
| Rank | 6 | 5 | 4 | 3 | 2 | 1 | |

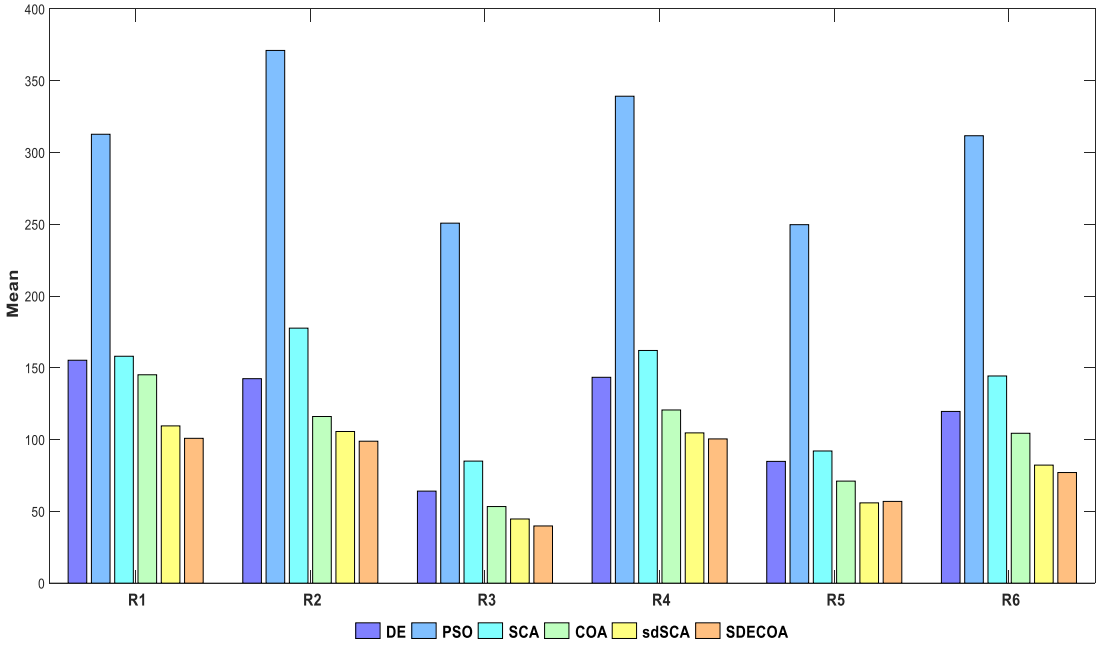


***Figure 8.*** *Average travel distance of each robot in Scenario 1.*

dependency on a single update strategy in COA. As a result, the algorithm is capable of achieving excellent performance across a variety of problem domains. Furthermore, the crossover probability, which was originally a constant parameter, is transformed into a dynamic adaptive variable to further improve the accuracy of the solutions. This approach effectively mitigates the premature convergence problem caused by the imbalance between local and global search capabilities in COA, thereby improving the algorithm's optimization capability.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table IX.*** *Average number of steps traveled by each robot in Scenario 1.*

| Robot | DE | PSO | SCA | COA | sdSCA | SDECOA |
|---|---|---|---|---|---|---|
| R1 | 129.45 | 250 | 129.65 | 122.05 | 88.1 | 89.9 |
| R2 | 114.6 | 288.8 | 147.6 | 96.6 | 86.65 | 90.35 |
| R3 | 53.15 | 195.4 | 66.15 | 43.45 | 37 | 35.05 |
| R4 | 116.65 | 281.4 | 133.35 | 97.6 | 87.65 | 90.9 |
| R5 | 67.7 | 200.2 | 75.7 | 56.1 | 46.3 | 48.95 |
| R6 | 97.85 | 256.5 | 116.35 | 87.15 | 68.55 | 68.35 |

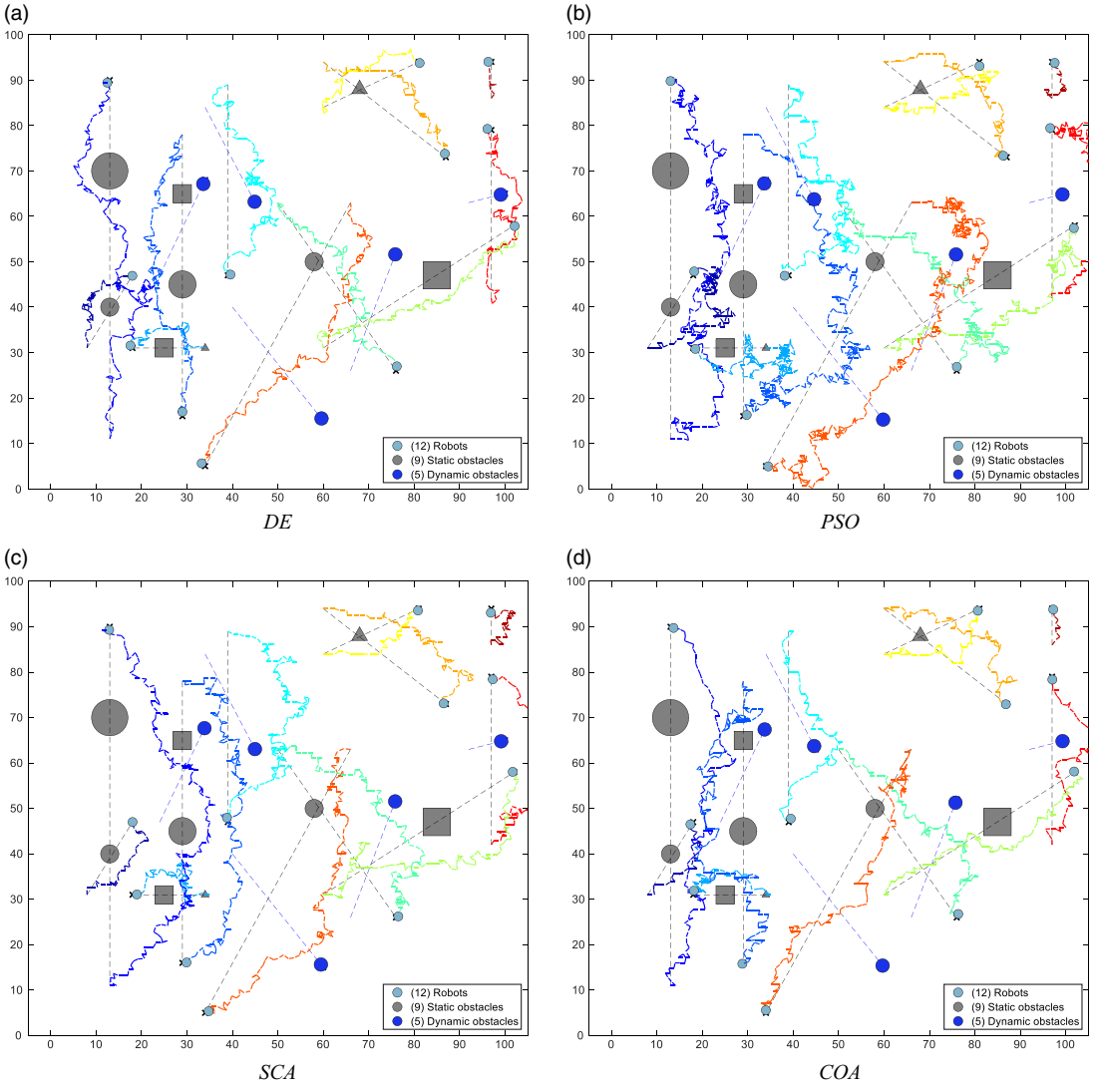


***Figure 9.*** *Paths of different algorithms in Scenario 2.*

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

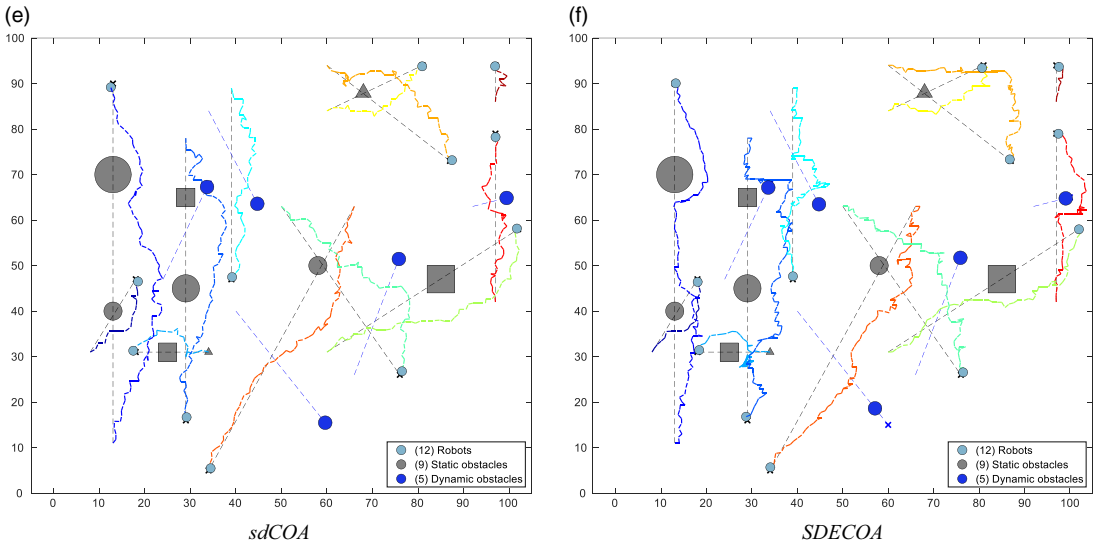


***Figure 9.*** *(Continued)*

***Table X.*** *Experimental results of the travel distance for each robot in Scenario 2.*

| Robot | Metric | DE | PSO | SCA | COA | sdSCA | SDECOA |
|---|---|---|---|---|---|---|---|
| R1 | Avg. | 62.787 | 217.614 | 63.358 | 45.436 | 26.614 | 31.861 |
| | SD. | 17.987 | 119.305 | 12.634 | 2.897 | 2.995 | 4.073 |
| R2 | Avg. | 164.269 | 431.117 | 213.012 | 184.073 | 121.577 | 97.637 |
| | SD. | 9.543 | 51.563 | 16.642 | 30.190 | 11.226 | 3.338 |
| R3 | Avg. | 250.674 | 330.084 | 173.439 | 201.653 | 85.770 | 105.505 |
| | SD. | 63.917 | 19.190 | 11.632 | 41.555 | 3.581 | 6.344 |
| R4 | Avg. | 52.440 | 229.792 | 71.882 | 77.764 | 32.838 | 29.979 |
| | SD. | 16.965 | 20.031 | 29.698 | 31.384 | 7.020 | 3.700 |
| R5 | Avg. | 111.945 | 310.797 | 114.428 | 110.812 | 60.319 | 64.881 |
| | SD. | 16.310 | 25.473 | 7.530 | 10.763 | 3.665 | 5.700 |
| R6 | Avg. | 122.856 | 188.722 | 106.414 | 99.016 | 62.420 | 65.306 |
| | SD. | 8.191 | 22.746 | 5.509 | 13.256 | 4.560 | 1.549 |
| R7 | Avg. | 108.073 | 165.359 | 116.620 | 92.062 | 67.966 | 58.992 |
| | SD. | 11.505 | 32.149 | 11.352 | 9.850 | 3.072 | 3.554 |
| R8 | Avg. | 68.966 | 115.199 | 62.237 | 47.225 | 37.932 | 32.271 |
| | SD. | 5.484 | 45.977 | 7.264 | 8.060 | 1.947 | 2.623 |
| R9 | Avg. | 89.652 | 148.906 | 84.867 | 76.711 | 51.414 | 52.491 |
| | SD. | 13.593 | 43.286 | 8.971 | 9.217 | 1.866 | 2.376 |
| R10 | Avg. | 155.862 | 425.183 | 176.663 | 355.010 | 88.339 | 97.291 |
| | SD. | 8.514 | 81.036 | 15.646 | 184.370 | 5.428 | 2.331 |
| R11 | Avg. | 81.344 | 254.331 | 110.142 | 86.818 | 54.585 | 47.961 |
| | SD. | 7.190 | 89.799 | 13.120 | 8.702 | 2.527 | 2.820 |
| R12 | Avg. | 20.528 | 113.579 | 41.494 | 20.356 | 11.243 | 11.153 |
| | SD. | 3.221 | 53.615 | 20.365 | 4.914 | 2.077 | 1.983 |
| Total mean distance | 1289.395 | 2930.682 | 1334.554 | 1396.936 | 701.017 | 695.327 | |
| Rank | 3 | 6 | 4 | 5 | 2 | 1 | |

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table XI.*** *Average number of steps traveled by each robot in Scenario 2.*

| Robot | DE | PSO | SCA | COA | sdSCA | SDECOA |
|---|---|---|---|---|---|---|
| R1 | 52 | 176.5 | 53.85 | 39 | 21.35 | 29.4 |
| R2 | 138.85 | 358.35 | 177.65 | 162.1 | 99.95 | 92.05 |
| R3 | 207.95 | 263.95 | 138.4 | 160.65 | 70.1 | 98.2 |
| R4 | 40.95 | 186.8 | 62.1 | 58.5 | 25.25 | 26.8 |
| R5 | 92.7 | 251.5 | 94.3 | 92.3 | 49.75 | 61.8 |
| R6 | 101.3 | 147.05 | 87.05 | 86.05 | 53.1 | 61.4 |
| R7 | 87.3 | 134.05 | 90.95 | 75.7 | 55.8 | 53.25 |
| R8 | 57.2 | 98.4 | 51.7 | 42.15 | 31 | 30.5 |
| R9 | 74.7 | 131.6 | 70.2 | 62.8 | 42.85 | 46.75 |
| R10 | 133.9 | 296.15 | 146.7 | 375.35 | 74.5 | 89.85 |
| R11 | 68.1 | 178.4 | 91.9 | 74.9 | 42.8 | 42.9 |
| R12 | 15.95 | 115.5 | 35.35 | 15.2 | 9.6 | 9.65 |

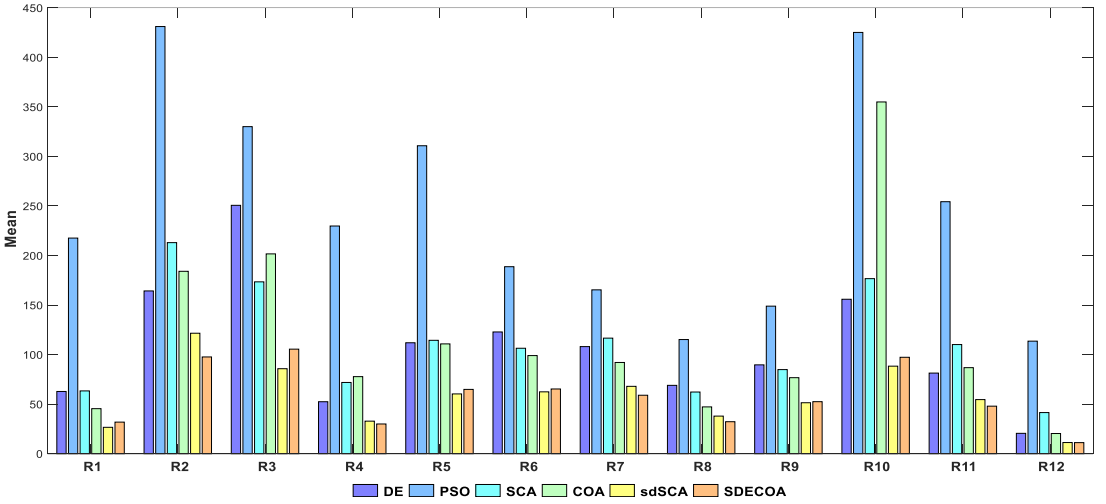


***Figure 10.*** *Average travel distance of each robot in Scenario 2.*

To validate the performance of the proposed algorithm, it was applied to the CEC2022 benchmark test suite and the CEC2020 real-world constrained optimization problems. Experimental results show that, compared to DE and COA, the proposed algorithm significantly improves performance, solving optimization problems even with over 100 constraints. When compared with the winning algorithm in the CEC2020 competition, SDECOA achieved the best results, as confirmed by the Friedman test. Finally, the algorithm was applied to multi-robot path planning problems in three complex scenarios. In Scenario 1, compared to the original DE and COA, the proposed algorithm reduced the average total travel distance of all robots by 33.2% and 22.4%, respectively, and reduced the average total movement steps by 31.1% and 21.6%. In Scenario 2, the average total travel distance was reduced by 46.1% and 50.2%, respectively, while the average total movement steps were reduced by 40.0% and 48.3%. In Scenario 3, compared to the original DE, the average total travel distance was reduced by 45.3%, and the average total movement steps were reduced by 32.7%. These experimental results demonstrate the strong competitiveness of the proposed algorithm.

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

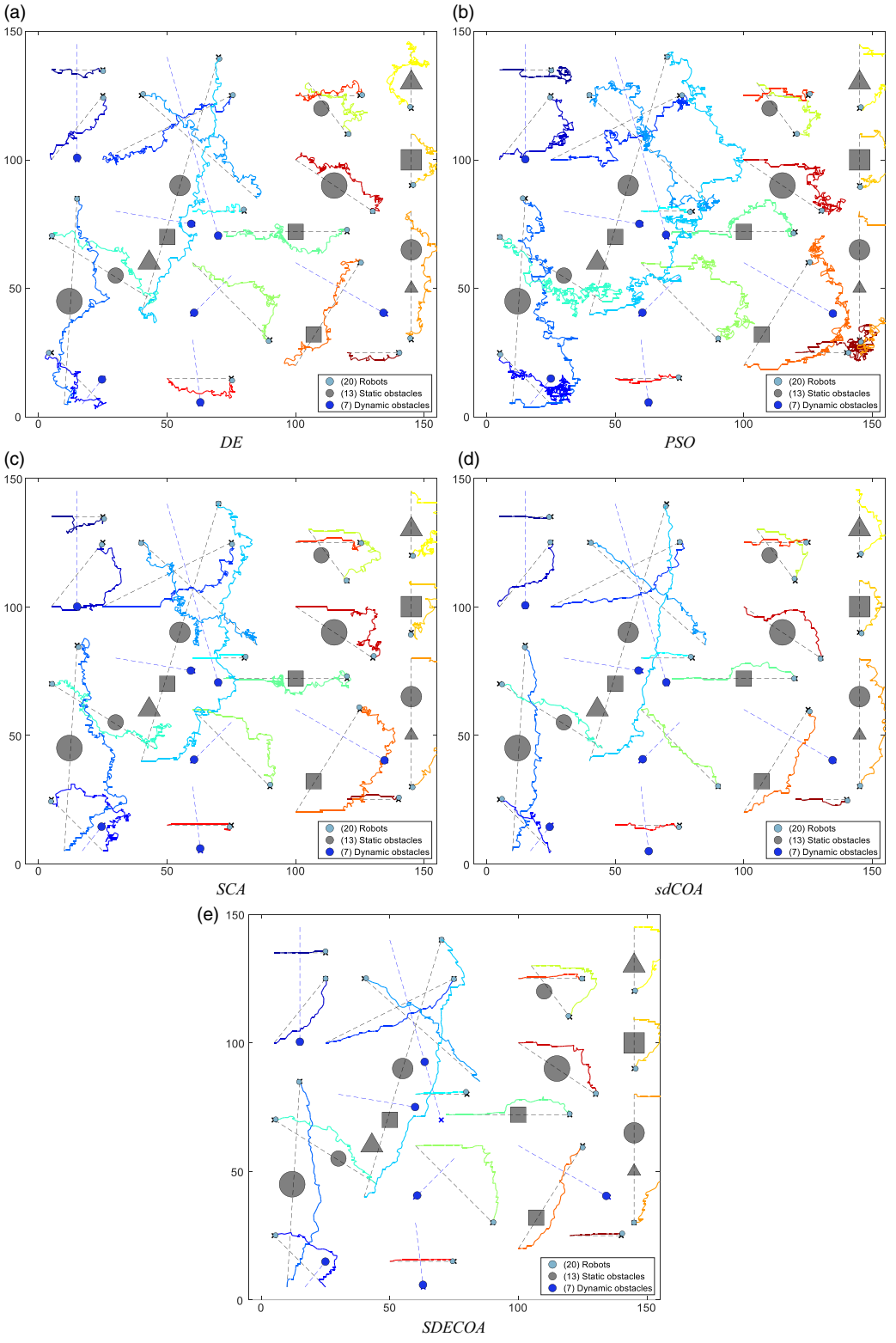


***Figure 11.*** *Paths of different algorithms in Scenario 3.*

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table XII.*** *Experimental results of the travel distance for each robot in Scenario 3.*

| Robot | Metric | DE | PSO | SCA | sdCOA | SDECOA |
|---|---|---|---|---|---|---|
| R1 | Avg. | 53.502 | 32.010 | 28.602 | 31.699 | 27.143 |
| | SD. | 2.190 | 7.675 | 4.651 | 7.293 | 3.153 |
| R2 | Avg. | 92.906 | 195.468 | 111.946 | 54.778 | 44.780 |
| | SD. | 8.406 | 38.925 | 10.642 | 2.342 | 2.796 |
| R3 | Avg. | 87.971 | 280.642 | 162.165 | 61.790 | 53.495 |
| | SD. | 13.500 | 42.123 | 22.905 | 3.785 | 5.882 |
| R4 | Avg. | 122.528 | 175.538 | 117.169 | 80.620 | 65.400 |
| | SD. | 13.259 | 17.704 | 10.251 | 5.467 | 1.827 |
| R5 | Avg. | 207.327 | 449.117 | 320.794 | 123.995 | 105.829 |
| | SD. | 3.290 | 40.807 | 62.608 | 2.335 | 3.750 |
| R6 | Avg. | 151.624 | 557.380 | 223.380 | 89.750 | 89.355 |
| | SD. | 6.048 | 27.258 | 21.816 | 4.122 | 5.880 |
| R7 | Avg. | 222.964 | 619.783 | 379.673 | 153.876 | 146.662 |
| | SD. | 13.297 | 19.180 | 7.887 | 5.124 | 12.084 |
| R8 | Avg. | 45.038 | 22.431 | 19.930 | 27.471 | 23.262 |
| | SD. | 1.543 | 1.281 | 0.594 | 3.355 | 2.120 |
| R9 | Avg. | 162.324 | 544.513 | 252.321 | 82.498 | 75.965 |
| | SD. | 22.005 | 34.009 | 43.137 | 6.072 | 15.959 |
| R10 | Avg. | 124.478 | 123.487 | 94.905 | 70.077 | 59.796 |
| | SD. | 9.392 | 9.636 | 15.047 | 5.913 | 2.866 |
| R11 | Avg. | 140.808 | 213.129 | 122.371 | 65.260 | 77.242 |
| | SD. | 5.820 | 17.069 | 27.072 | 1.813 | 8.708 |
| R12 | Avg. | 100.883 | 188.058 | 136.553 | 50.084 | 60.251 |
| | SD. | 17.664 | 48.471 | 14.059 | 6.638 | 8.156 |
| R13 | Avg. | 123.430 | 331.049 | 105.435 | 57.955 | 69.380 |
| | SD. | 10.472 | 134.419 | 9.614 | 4.339 | 3.409 |
| R14 | Avg. | 87.166 | 288.856 | 122.968 | 50.055 | 57.020 |
| | SD. | 16.001 | 65.006 | 24.217 | 8.777 | 6.060 |
| R15 | Avg. | 201.038 | 376.998 | 216.024 | 88.111 | 105.646 |
| | SD. | 8.560 | 7.395 | 52.682 | 3.847 | 9.627 |
| R16 | Avg. | 123.763 | 399.841 | 131.791 | 84.567 | 64.274 |
| | SD. | 13.694 | 82.944 | 10.701 | 5.227 | 2.397 |
| R17 | Avg. | 64.757 | 38.555 | 38.621 | 36.415 | 30.008 |
| | SD. | 12.230 | 7.315 | 3.008 | 1.580 | 2.495 |
| R18 | Avg. | 66.079 | 33.227 | 25.095 | 31.407 | 27.821 |
| | SD. | 7.634 | 4.176 | 0.821 | 4.405 | 2.736 |
| R19 | Avg. | 97.520 | 134.959 | 89.793 | 57.950 | 63.769 |
| | SD. | 17.077 | 29.111 | 7.381 | 5.984 | 9.539 |
| R20 | Avg. | 44.769 | 23.512 | 42.718 | 22.527 | 23.461 |
| | SD. | 8.028 | 1.906 | 12.308 | 1.497 | 2.663 |
| Total mean distance | 2320.876 | 5028.552 | 2742.255 | 1320.883 | 1270.558 | |
| Rank | 3 | 5 | 4 | 2 | 1 | |

This paper presents a novel solution approach for the multi-robot path planning problem, holding promise for addressing more complex path planning challenges in future environments, such as multi-robot path planning in three-dimensional spaces with multiple dynamic obstacles. According to the No Free Lunch theorem, no single algorithm can solve all problems, but continuous improvement can enable

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

***Table XIII.*** *Average number of steps traveled by each robot in Scenario 3.*

| Robot | DE | PSO | SCA | sdSCA | SDECOA |
|---|---|---|---|---|---|
| R1 | 43.2 | 25.5 | 21.75 | 23.55 | 27 |
| R2 | 77.1 | 155.45 | 87.7 | 43.85 | 41.15 |
| R3 | 76.5 | 240.7 | 138.05 | 52.15 | 51 |
| R4 | 102.2 | 149 | 96 | 64.8 | 61 |
| R5 | 170.7 | 376.05 | 275.1 | 104.15 | 98.4 |
| R6 | 127.8 | 450.4 | 185.6 | 74.4 | 83.95 |
| R7 | 180.85 | 512.5 | 315.85 | 128.55 | 141 |
| R8 | 36.3 | 17.4 | 15.45 | 22.55 | 22.75 |
| R9 | 135.45 | 441.65 | 213.15 | 65.3 | 79.45 |
| R10 | 97.45 | 95.85 | 79.4 | 55.4 | 55.35 |
| R11 | 115.6 | 173.1 | 115.15 | 52.6 | 72.45 |
| R12 | 79.35 | 161.7 | 109.8 | 40.8 | 55.8 |
| R13 | 99.5 | 260.6 | 89.25 | 47.55 | 63.35 |
| R14 | 74.9 | 261.05 | 101.45 | 39.05 | 55.3 |
| R15 | 168.1 | 310.55 | 193.1 | 73.1 | 101.3 |
| R16 | 102.4 | 333.25 | 107.6 | 69.5 | 59.8 |
| R17 | 55.95 | 29.4 | 31.4 | 28.9 | 28.75 |
| R18 | 58.55 | 27 | 20.45 | 25.6 | 26.8 |
| R19 | 77.7 | 108.85 | 75.5 | 48.45 | 60.95 |
| R20 | 35.7 | 18.5 | 35.8 | 18 | 21.65 |

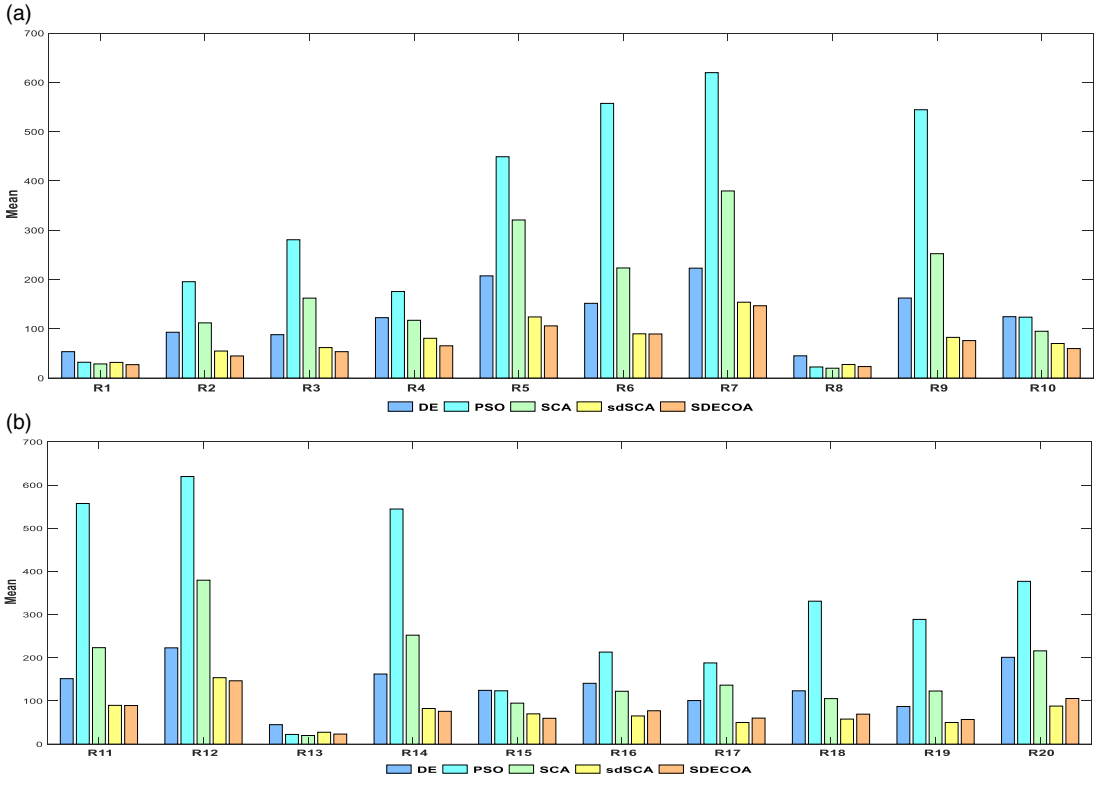


***Figure 12.*** *Average travel distance of each robot in Scenario 3.*

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

an algorithm to solve a broader range of problems. Of course, the number of position update strategies is not always a determining factor for success. In future research on intelligent algorithms, better position update strategies that align with the COA are expected to emerge, along with more challenging real-world optimization problems for SDECOA to solve. One limitation of the proposed algorithm is that it has a relatively large number of control parameters, and the effective update strategy at the initial stage may not necessarily be suitable for the later iterations. Future research could focus on applying the algorithm to more realistic path planning problems. Moreover, this paper only applies SDECOA to single-objective constrained optimization problems; future work could extend its application to high-dimensional multi-objective optimization problems.

**Author contributions.** Lun Zhu: Methodology and Writing – original draft. Guo Zhou: Writing – original draft. Yongquan Zhou: Supervisor and Writing – review and editing. Qifang Luo: Supervisor and Experimental results analysis. Huajuan Huang: Algorithm. Xiuxi Wei: Software and Results analysis.

**Data availability.** No data were used for the research described in the article.

**Financial support.** This work was supported by National Natural Science Foundation of China under Grant U21A20464, 62066005.

**Competing interests.** All the work outlined in this paper is our own except where otherwise acknowledged and referenced. The work contained in the manuscript has not been previously published, in whole or part, and is not under consideration by any other journal. All authors are aware of and accept responsibility for the Manuscript. The authors declare that they have no conflicts of interest.

**Ethical approval.** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

[1] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Autom. Control* **51**(3), 401–420 (2006).
[2] P. Tournassoud. A strategy for obstacle avoidance and its application to mullti-robot systems. **In:** *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, **3** (IEEE, 1986) pp. 1224–1229.
[3] T. Simeon, S. Leroy and J.-P. Lauumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Trans. Robot. Autom.* **18**(1), 42–49 (2002).
[4] E. Žunić, A. Djedović and B. Žunić, "Software solution for optimal planning of sales persons work based on Depth-First Search and Breadth-First Search algorithms," **In:** *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (IEEE, 2016) pp. 1248–1253.
[5] B. Fu, L. Chen, Y. Zhou, D. Zheng, Z. Wei, J. Dai and H. Pan, "An improved A* algorithm for the industrial robot path planning with high success rate and short length," *Robotics and Autonomous Systems* **106**, 26–37 (2018).
[6] P. Zhang, C. Xiong, W. Li, X. Du and C. Zhao, "Path planning for mobile robot based on modified rapidly exploring random tree method and neural network," *Int. J. Adv. Robot. Syst.* **15**(3), 1729881418784221 (2018).
[7] J. Kennedy and E. Russell. Particle swarm optimization. **In:** *Proceedings of ICNN 95-international conference on neural networks*, **4** (IEEE, 1995) pp. 1942–1948.
[8] M. Dorigo, B. Mauro and S. Thomas, "Ant colony optimization," *IEEE Comput. Intell. Mag.* **1**(4), 28–39 (2006).
[9] D. Karaboga and B. Bahriye, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optimiz.* **39**(3), 459–471 (2007).
[10] J. H. Holland, "Genetic algorithms," *Sci. Am.* **267**(1), 66–73 (1992).
[11] A. K. Qin, V. L. Huang and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.* **13**(2), 398–417 (2008).
[12] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, "GSA: A gravitational search algorithm," *Inform. Sci.* **179**(13), 2232–2248 (2009).
[13] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk and W. Al-Atabany, "Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems," *Appl. intell.* **51**(3), 1531–1551 (2021).
[14] M. Ghasemi, M. Zare, A. Zahedi, P. Trojovský, L. Abualigah and E. Trojovská, "Optimization based on performance of lungs in body: Lungs performance-based optimization (LPO)," *Comput. Method Appl. Mech. Eng.* **419**, 116582 (2024).
[15] Z. Tian and G. Mei, "Football team training algorithm: A novel sport-inspired meta-heuristic optimization algorithm for global optimization," *Expert Syst. Appl.* **245**, 123088 (2024).

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

[16] J. Geng, X. Sun, H. Wang, X. Bu, D. Liu, F. Li and Z. Zhao, "A modified adaptive sparrow search algorithm based on chaotic reverse learning and spiral search for global optimization," *Neural Comput. Appl.* **35**(35), 24603–24620 (2023).

[17] D. R. Parhi, "Chaos-based optimal path planning of humanoid robot using hybridized regression-gravity search algorithm in static and dynamic terrains," *Appl. Soft Comput.* **140**, 110236 (2023).

[18] Y. Li, J. Zhao, Z. Chen, G. Xiong and S. Liu, "A robot path planning method based on improved genetic algorithm and improved dynamic window approach," *Sustainability-Basel* **15**(5), 4656 (2023).

[19] L. Xu, C. Maoyong and S. Baoye, "A new approach to smooth path planning of mobile robot based on quartic Bezier transition curve and improved PSO algorithm," *Neurocomputing* **473**(2022), 98–106 (2022).

[20] M. Nazarahari, K. Esmaeel and D. Samira, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.* **115**, 106–120 (2019).

[21] T. W. Zhang, G. H. Xu, X. S. Zhan and T. Han, "A new hybrid algorithm for path planning of mobile robot," *J. Supercomput.* **78**(3), 4158–4181 (2022).

[22] X. Dai, S. Long, Z. Zhang and D. Gong, "Mobile robot path planning based on ant colony algorithm with A* heuristic method," *Front. Neurorobot.* **13**, 15 (2019).

[23] G. Chen and L. Jie, "Mobile robot path planning using ant colony algorithm and improved potential field method," *Comput. Intell. Neurosci.* **2019**(1), 1–10 (2019).

[24] Q. Zhang, X. Ning, Y. Li, R. G. l. Pan and L. Zhang, "Path planning of patrol robot based on modified grey wolf optimizer," *Robotica* **41**(7), 1947–1975 (2023).

[25] S. Liu, S. Liu and H. Xiao, "Improved gray wolf optimization algorithm integrating A* algorithm for path planning of mobile charging robots," *Robotica* **42**(2), 536–559 (2024).

[26] Y. Dai, S. Li, X. Chen, X. Nie, X. Rui and Q. Zhang, "Three-dimensional truss path planning of cellular robots based on improved sparrow algorithm," *Robotica* **42**(2), 347–366 (2024).

[27] S. J. Fusic and R. Sitharthan, "Self-adaptive learning particle swarm optimization-based path planning of mobile robot using 2D Lidar environment," *Robotica* **42**(4), 977–1000 (2024).

[28] M. Dehghani, Z. Montazeri, E. Trojovská and P. Trojovský, "Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems," *Knowl.-Based Syst.* **259**, 110011 (2023).

[29] F. A. Hashim, E. H. Houssein, R. R. Mostafa, A. G. Hussien and F. Helmy, "An efficient adaptive-mutated Coati optimization algorithm for feature selection and global optimization," *Alexandria Eng. J.* **85**, 29–48 (2023).

[30] E. Baş and Y. Gülnur, "Enhanced coati optimization algorithm for big data optimization problem," *Neural Process. Lett.* **55**(8), 10131–10199 (2023).

[31] E. Baş and Y. Gülnur, "A new binary coati optimization algorithm for binary optimization problems," *Neural Comput. Appl.* **36**(6), 2797–2834 (2024).

[32] H. M. Hasanien, I. Alsaleh, A. Alassaf and A. Alateeq, "Enhanced coati optimization algorithm-based optimal power flow including renewable energy uncertainties and electric vehicles," *Energy* **283**, 129069 (2023).

[33] H. Jia, S. Shi, D. Wu, H. Rao, J. Zhang and L. Abualigah, "Improve coati optimization algorithm for solving constrained engineering optimization problems," *J. Comput. Design Eng.* **10**(6), 2223–2250 (2023).

[34] A. Kumar, G. Wu, M. X. Ali, R. Mallipeddi, P. N. Suganthan and S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm Evol. Comput.* **56**, 100693 (2020).

[35] R. Akay and Y. Y. Mustafa, "Multi-strategy and self-adaptive differential sine-cosine algorithm for multi-robot path planning," *Expert Syst. Appl.* **232**, 120849 (2023).

[36] A. Draa, B. Samira and B. Imene, "A sinusoidal differential evolution algorithm for numerical optimization," *Appl. Soft Comput.* **27**, 99–126 (2015).

[37] S. Das and N. S. Ponnuthurai, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.* **15**(1), 4–31 (2010).

[38] S. Mirjalili, S. M. Mirjalili, A. Lewis and G. wolf optimizer, "Grey wolf optimizer," *Adv. Eng. Softw.* **69**, 46–61 (2014).

[39] Y. Wang, C. Zixing and Z. Qingfu, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.* **15**(1), 55–66 (2011).

[40] S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, "Optimization by simulated annealing," *Science* **220**(4598), 671–680 (1983).

[41] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowl.-based Syst.* **96**, 120–133 (2016).

[42] S. Mirjalili and L. Andrew, "The whale optimization algorithm," *Adv. Eng. Softw.* **95**, 51–67 (2016).

[43] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gen. Comput. Syst.* **97**, 849–872 (2019).

[44] J. Xue and S. Bo, "A novel swarm intelligence optimization approach: Sparrow search algorithm," *Syst. Sci. Control Eng.* **8**(1), 22–34 (2020).

[46] H. Song, J. Bei, H. Zhang, J. Wang and P. Zhang, "Hybrid algorithm of differential evolution and flower pollination for global optimization problems," *Expert Syst. Appl.* **237**, 121402 (2024).

[47] D. Sharma and S. D. Jabeen, "Hybridizing interval method with a heuristic for solving real-world constrained engineering optimization problems," *Structures* **56**, 104993 (2023).

[48] A. Kumar, S. Das and I. Zelinka. A self-adaptive spherical search algorithm for real-world constrained optimization problems. **In:** *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (2020) pp. 13–14.

[49] J. Gurrola-Ramos, A. Hernàndez-Aguirre and O. Dalmau-Cedeño, "COLSHADE for Real-World Single-Objective Constrained Optimization Problems," **In:** *2020 IEEE Congress On Evolutionary Computation (CEC)* (IEEE, 2020) pp. 1–8.


https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press

[50] A. Kumar, S. Das and I. Zelinka. A Modified Costandard Deviation Matrix Adaptation Evolution Strategy for Real-world Constrained Optimization Problems. **In:** *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion* (2020) pp. 8–12.
[51] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty and M. J. Ryan, "Multi-Operator Differential Evolution Algorithm for Solving Real-World Constrained Optimization Problems," **In:** *2020 IEEE Congress on Evolutionary Computation* (2020) pp. 1–8.
[52] M. Hellwig and H. G. Beyer, "A Modified Matrix Adaptation Evolution Strategy with Restarts for Constrained Real-World Problems," **In:** *2020 IEEE Congress on Evolutionary Computation (CEC)* (IEEE, 2020) pp. 1–8.
[53] J. Xu and X. Lihong, "Optimal stochastic process optimizer: A new metaheuristic algorithm with adaptive exploration-exploitation property," *IEEE Access.* **9**, 108640–108664 (2021).
[54] A. E. S. Ezugwu, A. O. Adewumi and M. E. Frîncu, "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem," *Expert Syst. Appl.* **77**, 189–210 (2017).

**Cite this article:** L. Zhu, G. Zhou, Y. Zhou, Q. Luo, H. Huang and X. Wei, "Self-adaptive differential evolution-based coati optimization algorithm for multi-robot path planning", Robotica. https://doi.org/10.1017/S0263574725000049

https://doi.org/10.1017/S0263574725000049 Published online by Cambridge University Press