Journal of Systems Engineering and Electronics
Vol. 37, No. 3, June 2026, pp.878 – 896

Odd-even dimension RUNge Kutta optimization algorithm and its application

WANG Lin, PI Yingying, and WANG Xuerui*

School of Management, Huazhong University of Science and Technology, Wuhan 430074, China

Abstract: This research proposes an odd-even dimension RUNge Kutta algorithm (ODRUN) to solve global optimization and a well-known NP-hard problem in inventory management. The rpoposed algorithm integrates odd-even dimensional, fourth-order Runge-Kutta method, and neighbor search strategies. This hybrid approach significantly improves population diversity, avoids local optima, and enhances convergence accuracy. To validate the performance of the proposed algorithm, a widely recognized benchmark function suit from CEC2022 is first employed. Results confirm that ODRUN achieves an overall effectiveness ratio of 66.67% across three statistical indicators (best, mean, and standard deviation) for 12 benchmark functions. The test shows this algorithm is ranked first compared to seven state-of-the-art metaheuristic algorithms. Furthermore, ODRUN is applied to the joint replenishment problem with imperfect items and trade credit. Numerical examples from 600 randomly generated large-scale instances highlight that the algorithm's performance remains unaffected by an increase in problem scale. The significant cost savings brought by the ODRUN algorithm, with the maximum improvement ratio in average cost and best-found total cost ranging from 14.81% to 19.5%, are achieved in comparison to other algorithms. In conclusion, ODRUN is an effective and robust tool for complex optimization problems.

Keywords: metaheturitics, RUNge Kutta algorithm, odd-even dimension search strategy, neighbor search strategy, joint replenishment problem with imperfect items and trade credit.

DOI: 10.23919/JSEE.2026.000115

1. Introduction

Meta-heuristic algorithms, known for their gradient-free mechanism and flexibility, are widely used in engineering, manufacturing, and business optimization problems, such as genetic algorithm (GA) [1], differential evolution (DE) [2], grey wolf optimizer (GWO) [3], and walrus optimize (WO) [4]. Among them, a population-based RUNge Kutta algorithm (RUN) proposed by Ahmadianfar et al. [5] has gained attention for its mathematical features of the Runge-Kutta method, which efficiently solves ordinary differential equations without requiring high-order derivatives. The search mechanism and enhanced solution quality mechanism of the algorithm are designed to balance the exploration and exploitation phases and to speed up convergence. The algorithm has achieved great performance in different fields, such as constrained engineering problems [6], wind power forecasting [7], feature selection problems [8,9], hyperparameter adjusting for medical diagnosis [10], and parameter identification [11,12].

Despite its effectiveness, the RUN algorithm often suffers from local optima and premature convergence, limiting its performance in highly complex problems. To enhance the performance of RUN, scholars have explored various improved versions. These improvements can be categorized into four main approaches. First, improvements in population initialization have been explored. For example, Yildiz et al. [6] explored the integration of ten different chaotic mappings with the RUN algorithm to enhance population diversity. Second, hybridization with advanced learning strategies has been applied to strengthen the algorithm's convergence and exploration. Zhang et al. [9] introduced an improved version based on opposition-based learning and cuckoo search, which accelerates convergence and enhances the performance of exploration. Third, researchers have proposed innovative candidate solution generation mechanisms. El-Sattar et al. [13] provided a new tool employing a quantum mechanism to generate new candidate solutions. Nassef et al. [14] proposed a modified RUN based on the orthogonal learning strategy to generate new candidate solutions. Finally, multi-strategy integration has been used to enhance the robustness and adaptability of the algorithm. Chen et al. [15] developed a novel strategy integrating multiple approaches to generate candidate solutions.

Manuscript received May 30, 2024.
*Corresponding author.
This work was supported by the National Social Science Foundation of China (20&ZD126).

Motivated by the no free lunch (NFL) theorem [16], which highlights the impossibility of a universal optimization algorithm, this study aims to further enhance the RUN algorithm by integrating novel strategies for global optimization.

This paper applies two strategies to improve the performance of RUN in solving global optimization problems. First, an update mechanism integrating the odd-even dimension search strategy with the fourth-order Runge-Kutta method is proposed to improve the population diversity and enhance the global search capability. By taking advantage of dimensional information and slope variations computed by the fourth-order Runge-Kutta method, this method exhibits high computational accuracy and enhances diversity, facilitating the expansion of the search space, escaping local optima, and providing more diverse update paths for the population. This expansion strengthens the algorithm's exploration ability and global optimization performance. Second, a modified enhanced solution quality scheme based on a neighbor search strategy is proposed to improve the accuracy of the optimal solution. This scheme enhances candidate solution quality through local search, ensuring better convergence and more reliable optimization results. To validate the effectiveness of odd-even dimension RUN (ODRUN), it was tested on 12 benchmark functions from CEC2022, involving a range of global optimization scenarios. Key statistical metrics, such as mean, standard deviation, best-fitness values, and rank were analyzed, and the overall performance of ODRUN was compared with seven other state-of-the-art algorithms, including coati optimization algorithm (COA) [17], GA, DE, gradient-based optimizer (GBO) [18], WO, GWO, and original RUN algorithm. Results show that ODRUN has excellent convergence accuracy and stability. Furthermore, ODRUN was applied to solve the joint replenishment problem with imperfect items and trade credit (IJRPTC), which is a discrete mixed-integer nonlinear NP-hard problem with mathematical significance. The major contributions are as follows:

(i) An update mechanism that integrates the odd-even dimension search strategy with the fourth-order Runge-Kutta method is proposed to enhance the population diversity and global search capability of ODRUN.

(ii) A neighbor search strategy is introduced to improve the convergence accuracy, with ODRUN achieving an overall effectiveness ratio of 66.67% on CEC2022 benchmark functions.

(iii) The proposed algorithm achieves an effectiveness rate of 93.34% in solving the IJRPTC problem, with improvements in average cost and best-found total cost ranging from 14.81% to 19.5% compared to other algorithms, based on 600 randomly generated instances.

The paper is structured as follows: Section 2 introduces the proposed ODRUN algorithm; Section 3 presents Experiment 1, which evaluates the proposed algorithm on a widely recognized benchmark function suite from CEC2022; Section 4 discusses Experiment 2, which evaluates the algorithm's performance on the well-known NP-hard IJRPTC problem, and Section 5 outlines the conclusions and future research.

2. ODRUN

This study proposes ODRUN to solve complex optimization problems. The ODRUN utilizes the odd-even dimension search strategy and RK4 mechanism to update the solution, and a modified enhanced solution quality scheme to improve the precision of the optimal solution. The following sections outline the core strategy and the procedure of ODRUN.

2.1 The fourth-order Runge-Kutta search mechanism

The fourth-order Runge-Kutta method is the root of RUN [5], the core search mechanism is as follows:

$$\mathbf{SM} = \frac{1}{6} \times \boldsymbol{X}_{\mathrm{RK}} \times \Delta \boldsymbol{X} \tag{1}$$

where **SM** represents the search mechanism, $\boldsymbol{X}_{\mathrm{RK}}$ represents the fourth-order Runge-Kutta method, determined by four parameters.

$$\boldsymbol{X}_{\mathrm{RK}} = \boldsymbol{\lambda}_1 + 2 \times \boldsymbol{\lambda}_2 + 3 \times \boldsymbol{\lambda}_3 + \boldsymbol{\lambda}_4, \tag{2}$$

$$\boldsymbol{\lambda}_1 = \frac{1}{2\Delta \boldsymbol{X}} (\mathbf{rand} \times \boldsymbol{X}_{\mathrm{worst}} - \boldsymbol{u} \times \boldsymbol{X}_{\mathrm{better}}), \tag{3}$$

$$\boldsymbol{\lambda}_2 = \frac{1}{2\Delta \boldsymbol{X}} (\mathbf{rand} \times (\boldsymbol{X}_{\mathrm{worst}} + \mathbf{rand}_1 \times \boldsymbol{\lambda}_1 \times \Delta \boldsymbol{X}) - (\boldsymbol{u} \times \boldsymbol{X}_{\mathrm{better}} + \mathbf{rand}_2 \times \boldsymbol{\lambda}_2 \times \Delta \boldsymbol{X})), \tag{4}$$

$$\boldsymbol{\lambda}_3 = \frac{1}{2\Delta \boldsymbol{X}} \left(\mathbf{rand} \times (\boldsymbol{X}_{\mathrm{worst}} + \mathbf{rand}_1 \times \left(\frac{1}{2} \boldsymbol{\lambda}_2 \right) \times \Delta \boldsymbol{X}) - (\boldsymbol{u} \times \boldsymbol{X}_{\mathrm{better}} + \mathbf{rand}_2 \times \boldsymbol{\lambda}_2 \times \Delta \boldsymbol{X}) \right), \tag{5}$$

$$\boldsymbol{\lambda}_4 = \frac{1}{2\Delta \boldsymbol{X}} (\mathbf{rand} \times (\boldsymbol{X}_{\mathrm{worst}} + \mathbf{rand}_1 \times \boldsymbol{\lambda}_3 \times \Delta \boldsymbol{X}) - (\boldsymbol{u} \times \boldsymbol{X}_{\mathrm{better}} + \mathbf{rand}_2 \times \boldsymbol{\lambda}_3 \times \Delta \boldsymbol{X})), \tag{6}$$

$$\boldsymbol{u} = \mathrm{round}((\mathbf{1} + \mathbf{rand}) \times (\mathbf{1} - \mathbf{rand})), \tag{7}$$

where **rand**, $\mathbf{rand}_1$, and $\mathbf{rand}_2$ are random vectors uniformly distributed in the interval [0,1]; $\boldsymbol{X}_{\mathrm{better}}$ and $\boldsymbol{X}_{\mathrm{worst}}$ denote the best and worst solutions of three random agents $(\boldsymbol{X}_{r1}, \boldsymbol{X}_{r2}, \boldsymbol{X}_{r3})$, and $r1 \neq r2 \neq r3 \neq l$; $\boldsymbol{u}$ is a random vector; $\Delta \boldsymbol{X}$ is position increment; **Stp** is the step size.

The following is how it works:

$$\Delta \boldsymbol{X} = 2 \times \mathbf{rand} \times |\mathbf{Stp}|, \tag{8}$$

$$\mathbf{Stp} = \mathbf{rand} \times ((\boldsymbol{X}_{\text{better}} - \mathbf{rand} \times \boldsymbol{X}_{\text{avg}}) + \boldsymbol{\gamma}), \tag{9}$$

$$\boldsymbol{\gamma} = \mathbf{rand} \times \left(\boldsymbol{X}_l - \mathbf{rand} \times (\text{ub} - \text{lb}) \times \exp\left(-4 \times \frac{\text{it}}{\text{Maxit}} \right) \right), \tag{10}$$

where $\boldsymbol{X}_{\text{avg}}$ is the average of all agents; ub and lb are the upper and lower bounds of agents; it is the current iteration number; Maxit is the maximum number of iterations; $\boldsymbol{\gamma}$ is a scaling factor. $\boldsymbol{X}_{bi}$ is the best position of the three possible random agents $(\boldsymbol{X}_{r1}, \boldsymbol{X}_{r2}, \boldsymbol{X}_{r3})$; $\boldsymbol{X}_{\text{worst}}$ and $\boldsymbol{X}_{\text{better}}$ are calculated as follows:

$$\begin{aligned} &\text{if fitness}(\boldsymbol{X}_l) < \text{fitness}(\boldsymbol{X}_{bi}), \\ &\quad \boldsymbol{X}_{\text{better}} = \boldsymbol{X}_l, \\ &\quad \boldsymbol{X}_{\text{worst}} = \boldsymbol{X}_{bi}, \\ &\quad \text{else} \\ &\quad \boldsymbol{X}_{\text{better}} = \boldsymbol{X}_{bi}, \\ &\quad \boldsymbol{X}_{\text{worst}} = \boldsymbol{X}_l, \end{aligned}$$

end.

2.2 Odd-even dimension search strategy

An odd-even dimension search strategy is developed in ODRUN. This strategy utilizes the information across different dimensions to generate new candidate solutions, which significantly improves population diversity and enhances the algorithm's global search capability. By utilizing differentiated information between dimensions, the strategy enables the population to explore a wider range of potential solutions, thus preventing the algorithm from getting stuck in local optima. It overcomes the shortcoming of insufficient global search that may result in sometimes only local optimal solutions being found. The following is how it works:

$$X_{\text{new},d} = \begin{cases} X_{\tilde{l},\tilde{J}} + \left(X_{\tilde{r}l,\text{array}(1)} - X_{\tilde{l},\text{array}(r\tilde{J})} \right) \cdot \\ \quad (1 + \sin(2\pi \cdot \text{rand}_1)) \cdot \\ \quad \sin(2\pi \cdot \text{rand}_2), \ d \text{ is odd} \\ X_{\tilde{l},\tilde{J}} + \left(X_{\tilde{r}l,\text{array}(1)} - X_{\tilde{l},\text{array}(r\tilde{J})} \right) \cdot \\ \quad (1 + \cos(2\pi \cdot \text{rand}_3)) \cdot \\ \quad \cos(2\pi \cdot \text{rand}_4), \ d \text{ is even} \end{cases} \tag{11}$$

where $X_{\tilde{l},\tilde{J}}$ denotes the value of the current solution $\boldsymbol{X}_{\tilde{l}}$ in the $\tilde{J}$th dimension; $\boldsymbol{X}_{\tilde{r}_1}(\tilde{r}_1 \neq \tilde{l})$ is the randomly selected value of another individual from the population; rand_1, rand_2, rand_3, and rand_4 are random numbers in the range (0,1) that increase randomness; $\text{array} = \text{randperm}(\text{dim})$ generates a random arrangement from 1 to dim to select the dimensions to mutate; $X_{\tilde{r}l,\text{array}(1)} - X_{\tilde{l},\text{array}(r\tilde{J})}$ represents the difference between two solutions in the randomly selected dimensions. For odd dimensions, the sine function is utilized to randomly increase or reduce the difference, while for even dimensions, the cosine function is employed to achieve a similar modification. The $X_{\text{new},d}$ updates the values of individuals for both odd and even dimensions in a nonlinear manner, resulting in new hybrid random individuals. This approach helps the algorithm avoid local optima, increases population diversity, and searches for a greater solution space. By exchanging information with other agents and dynamically adjusting (either increasing or weakening) the information using sine and cosine functions, the strategy strengthens the global search ability and maintains the population's diversity during the process.

2.3 Neighbor search strategy

The ODRUN employs a neighbor search strategy to replace the original enhanced solution quality scheme. This strategy searches for better solutions by continuously interacting with neighbors in the problem space. The core idea is to facilitate global exploration through local search and information sharing, thus avoiding local optima and improving the solution accuracy in the later stages.

Firstly, we determine the neighbor range by

$$\text{radius} = \lambda \cdot \|\text{ub} - \text{lb}\| \tag{12}$$

where λ is the coefficient that controls the search range, $\|\cdot\|$ represents the Euclidean distance between the upper and lower bounds of the search space. Based on it, agents whose Euclidean distance from $\boldsymbol{X}_{\text{new1}}$ is less than radius are neighbors of $\boldsymbol{X}_{\text{new1}}$. Once the neighbors are identified, the neighbor agents of $\boldsymbol{X}_{\text{new1}}$ share information with $\boldsymbol{X}_l$ by

$$X_{NS,\tilde{J}} = X_{\text{new1},\tilde{J}} + \text{rand}_{\tilde{J}} \cdot (X_{\tilde{r}_2,\tilde{J}} - X_{\text{new1},\tilde{J}}) \tag{13}$$

where $X_{\tilde{r}_2,\tilde{J}}$ denotes the $\tilde{J}$th dimension of the random neighbor agent of $\boldsymbol{X}_l$, and $\text{rand}_{\tilde{J}}$ is a random number. This process updates the solution based on the information from neighboring agents.

2.4 Procedure of ODRUN

2.4.1 Initialization

As a population metaheuristic, ODRUN randomly generates l populations in the initial stage. The initial individual is as follows:

$$X_{\tilde{l},\tilde{J}} = \text{lb} + \text{rand}\,(\text{ub} - \text{lb}) \tag{14}$$

where $\tilde{l} = 1,2,\cdots,L$, $\tilde{J} = 1,2,\cdots,\text{dim}$, ub and lb are the upper and lower bounds of the variable, and $\text{rand} \in [0,1]$.

2.4.2 Hybrid updating solution

The improved updating mechanism uses the fourth-order Runge-Kutta search mechanism and an odd-even dimension search strategy. The following is how it works:

$$\boldsymbol{x}_{\text{new1}}=\begin{cases}\boldsymbol{X}_{\text{new}}+\text{SF}_{\bar{l}}\times\mathbf{SM}, & \text{af}>1\\ \boldsymbol{X}_m+\boldsymbol{r}\times\text{SF}_{\bar{l}}\times g\times\boldsymbol{X}_m+ & \\ \quad \text{SF}_{\bar{l}}\times\mathbf{SM}+\mathbf{mu}\times(\boldsymbol{X}_{r1}-\boldsymbol{X}_{r2}), & \text{else}\end{cases} \tag{15}$$

in which

$$\theta=2\times\left(1-\frac{\text{it}}{\text{Maxit}}\right), \tag{16}$$

$$\text{af}=2\lg\left(\frac{1}{\text{rand}}\right)\theta. \tag{17}$$

According to the rule in (15), ODRUN selects either the global exploration phase or local exploitation phase based on an adaptive factor (af). With the increase in the number of iterations, the value of af decreases gradually and exhibits fluctuations, directing the algorithm from exploration to exploitation. When af > 1, the new candidate solution is generated based on $\boldsymbol{X}_{\text{new1}}$, otherwise, a solution is generated based on $\boldsymbol{X}_m$. The candidate solution $\boldsymbol{X}_{\text{new}}$ is generated using the odd-even dimension strategy, which updates the values of the odd and even dimensions independently. This enhances flexibility in updating the individuals and increases the diversity of the population. $\boldsymbol{X}_{\text{new}}$ is generated by (11), $\boldsymbol{X}_m$ is calculated as follows:

$$\boldsymbol{X}_m=\text{rand}_5\times\boldsymbol{X}_{\text{best}}+(1-\text{rand}_5)\times\boldsymbol{X}_{\text{lbest}} \tag{18}$$

where rand_5 is a random number in the range (0,1), which introduces randomness to prevent the algorithm from falling into local optima. Here, $\boldsymbol{X}_{\text{best}}$ represents the best-so-far solution, and $\boldsymbol{X}_{\text{lbest}}$ is the best solution found in the current iteration. The difference between $\boldsymbol{X}_{\text{new}}$ and $\boldsymbol{X}_m$ is that $\boldsymbol{X}_{\text{new}}$ focuses on utilizing new information to explore new regions, while $\boldsymbol{X}_m$ emphasizes known information to search around existing solutions. This helps to achieve a balance between exploration and exploitation. $\boldsymbol{X}_{r1}$ and $\boldsymbol{X}_{r2}$ are two random solutions. g is a random number within the range of [0,2]. $\mathbf{mu}=0.5+0.1\times\mathbf{randn}$, **randn** is random vector with a normal distribution, **SF** refers to the search mechanism which is impacted by two hyperparameters a and b. **SF** is calculated as follows:

$$\mathbf{SF}=2(\mathbf{0.5}-\mathbf{rand})\times f, \tag{19}$$

$$f=a\cdot\exp\left(-b\cdot\text{rand}\cdot\left(\frac{\text{it}}{\text{Maxiter}}\right)\right). \tag{20}$$

Finally, the fitness of $\boldsymbol{X}_{\text{new1}}$ and the current solution are evaluated. The algorithm saves the solution with better fitness and continues the search with it.

2.4.3 Modified enhanced solution quality

To enhance the solution quality, ODRUN employs a novel neighbor search strategy that replaces the original enhanced solution quality scheme. This strategy iteratively searches for better solutions by exchanging information with neighboring agents in the solution space. The following is how it works:

$$\boldsymbol{X}_{\text{new2}}=\boldsymbol{X}_{NS}. \tag{21}$$

In summary, the ODRUN algorithm starts with a randomly initialized population. During the search process, it generates candidate solutions by integrating the fourth-order Runge-Kutta method with the odd-even dimension strategy. The quality of these solutions is then enhanced through a neighborhood strategy. The pseudo-code of ODRUN is presented in Algorithm 1.

Algorithm 1 Pseudo-code of ODRUN

1. Stage 1. Initialization
2. Set the algorithm parameters L, dim, and Maxit
3. Generate the initial position $\boldsymbol{X}_l(l=1,2,\cdots,L)$
4. Calculate the fitness value of each position fitness$(\boldsymbol{X}_l)$
5. Save the historical optimal position $\boldsymbol{X}_{\text{best}}$ and current optimal position $\boldsymbol{X}_{\text{lbest}}$
6. Stage 2. ODRUN operators
7. For it = 1 : Maxit
8. For l = 1 : L
9. Calculate $\boldsymbol{X}_{\text{new1}}$ by (15)
10. End For
11. Modified Enhance the solution quality
12. Calculate $\boldsymbol{X}_{\text{new2}}$ by (21)
13. Update positions $\boldsymbol{X}_{\text{best}}$ and $\boldsymbol{X}_{\text{lbest}}$
14. it = it + 1
15. End For
16. Stage 3. Return $\boldsymbol{X}_{\text{best}}$

2.5 Complexity analysis of ODRUN

The computational complexity of the metaheuristic algorithm evaluates the runtime of the algorithm. Suppose the population size of the ODRUN is NP, the maximum number of iterations is Maxit, and the dimension of the decision variable is Maxit. Therefore, the computational complexity of the initialization phase is O(NP), the evaluation of the fitness is $O(\text{NP}\times\text{Maxit})$ the solution positions updating is $O(\text{NP}\times\text{Maxit}\times\text{dim})$, and the modified enhanced solution quality is $O(\text{NP}\times\text{Maxit}\times\text{dim})$. As a consequence, the complexity of ODRUN is O(NP+ $\text{NP}\times\text{Maxit}\times(1+2\times\text{dim})$).

3. Experiment 1: applying ODRUN for solving global optimization of benchmark problems

The ODRUN algorithm was verified using 12 benchmark functions from the IEEE Congress on Evolutionary Computation (CEC'2022) [19]. The set of benchmark functions involves four families of mathematical functions: unimodal functions (C_1^{22}), multimodal functions (C_2^{22}–C_5^{22}), and hybrid functions (C_6^{22}–C_8^{22}), and composition Functions (C_9^{22}–C_{12}^{22}). Some well-known algorithms, including COA, GA, DE, GBO, WO, GWO, and the original RUN algorithm. These algorithms are selected for comparison primarily due to their proven effectiveness in solving a wide range of optimization problems. Additionally, they represent both classical and recent state-of-the-art intelligent algorithms, providing a stable benchmark for the reliable evaluation of new algorithms.

3.1 Experiment setting

To ensure fairness and comparability, each algorithm was run 30 times for each benchmark function, with 1 000 iterations per run and a population size of 30. The parameter settings for each algorithm are summarized in Table 1. These parameters are set to their default values, as recommended in the corresponding works (see [1,2,3,4,17,20,21]). For COA, GA, DE, GBO, WO, and RUN, the control parameters are set according to their original work.

Table 1 Parameter setting of algorithms

Algorithm	Parameter
COA	None
GA	$F = 0.7$; $C_r = 0.3$
DE	$P_c = 0.9$; $P_m = 0.5$
GBO	$\beta_{\min} = 0.2$ $\beta_{\max} = 1.2$, $pr = 0.5$
WO	$P = 0.4$
GWO	$a_{\min} = 0$; $a_{\max} = 2$
RUN	$a = 20$; $b = 12$
ODRUN	$a = 20$; $b = 16$

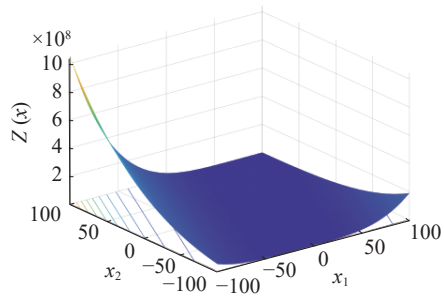
All experimental series are conducted on simulation software installed on Windows 11 (64-bit) operating system with an Intel Core(TM) i9-12900H, 2.5 GHz CPU, and 16 GB RAM. Quantitative measures (best, mean, and standard deviation values of obtained fitness) are presented in Table 2, qualitative measures (three dimensional (3-D) map for two dimensional (2-D) function, search history, population diversity, convergence curve) are displayed in Fig. 1, and the comparisons of convergence curves are shown in Fig. 2, and the comparisons of boxplot graphs of algorithms are provided in Fig. 3.

Table 2 A comparison of the statistical measures on CEC2022 test functions

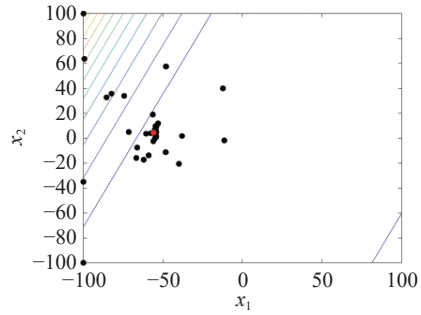
Function	Indicator	COA	GA	DE	GBO	WO	GWO	RUN	ODRUN
C_1^{22}	Best	2 757.61	13 543.42	34 495.85	300.00	301.17	340.71	300.00	300.00
	Mean	8 847.49	25 304.91	77 328.52	300.00	312.77	1 644.17	300.00	300.00
	Std	3 401.75	15 086.33	14 110.44	1.92E−08	9.89	1 668.50	3.00E−03	6.63E−08
C_2^{22}	Best	469.75	658.02	828.93	400.00	400.65	407.13	400.00	400.11
	Mean	1 588.77	1 388.33	1 323.65	405.00	429.08	425.65	403.89	406.66
	Std	814.04	514.02	387.82	3.16	33.57	22.00	4.47	3.04
C_3^{22}	Best	625.28	655.73	686.58	600.00	600.04	600.05	603.05	600.00
	Mean	648.79	660.77	704.95	600.89	600.78	601.21	614.61	600.07
	Std	9.88	4.50	9.80	2.95	0.75	1.69	6.75	0.14
C_4^{22}	Best	834.37	872.53	907.16	810.94	808.06	803.98	812.93	801.99
	Mean	851.23	886.71	933.83	822.15	833.91	815.96	821.92	809.29
	Std	8.86	11.83	17.19	8.27	14.08	8.26	5.08	4.18
C_5^{22}	Best	1 021.00	2 128.97	2 467.22	900.09	900.00	900.02	933.47	900.00
	Mean	1 426.36	2 197.37	4 896.78	908.20	904.37	910.98	1 010.62	900.40
	Std	199.24	205.42	1 352.10	11.00	4.88	19.59	46.63	0.41
C_6^{22}	Best	4.42E+05	1.48E+08	1.38E+08	1 803.36	1 870.65	2 016.12	1 877.82	1 835.92
	Mean	7.73E+07	1.40E+09	1.66E+09	2 788.76	3 394.99	5 605.32	3 378.26	3 190.18
	Std	8.28E+07	6.81E+08	1.12E+09	1 611.25	1 722.46	2 343.16	1 016.88	1 555.41

Continued

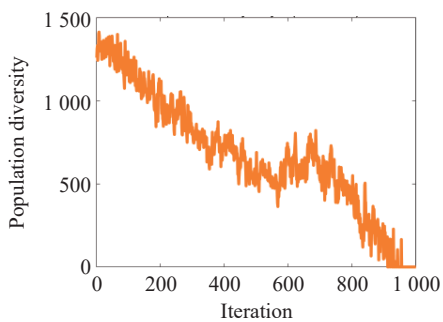
Function	Indicator	COA	GA	DE	GBO	WO	GWO	RUN	ODRUN
C_7^{22}	Best	2053.61	2114.07	2120.44	2000.99	2020.07	2001.96	2024.98	2000.06
	Mean	2086.42	2251.26	2276.96	2021.79	2031.72	2026.91	2043.96	2015.94
	Std	19.89	61.35	61.76	8.93	14.79	12.63	13.11	8.53
C_8^{22}	Best	2225.34	2269.30	2326.72	2200.86	2207.66	2207.04	2220.70	2200.49
	Mean	2249.37	3824.19	3378.14	2220.23	2224.19	2227.52	2223.38	2214.27
	Std	26.87	5009.21	2627.85	4.64	5.04	22.39	1.61	9.07
C_9^{22}	Best	2661.42	2764.29	2679.05	2529.28	2529.28	2529.29	2529.28	2529.28
	Mean	2754.82	3115.66	3168.97	2534.18	2534.12	2571.68	2539.08	2529.28
	Std	58.90	229.40	274.86	26.83	10.55	39.98	37.28	0.00
C_{10}^{22}	Best	2526.98	2517.95	2759.66	2500.32	2500.49	2500.31	2500.62	2500.18
	Mean	2781.64	2638.22	3950.97	2537.27	2558.58	2568.84	2555.17	2500.33
	Std	362.03	132.44	542.04	57.26	63.85	56.98	59.22	0.08
C_{11}^{22}	Best	2983.33	5266.45	38417.93	2600.00	2600.07	2720.07	2600.01	2600.00
	Mean	3982.92	5267.08	84468.75	2753.79	2701.68	2983.52	2703.79	2600.00
	Std	332.46	0.33	19472.43	152.38	181.12	132.56	150.26	1.45E−04
C_{12}^{22}	Best	2892.11	2880.28	2872.04	2861.40	2859.37	2859.56	2859.37	2859.37
	Mean	2973.51	2931.93	2882.94	2864.56	2863.87	2865.68	2864.51	2862.52
	Std	51.25	25.55	6.11	1.73	1.71	5.81	1.39	1.25
Friedman mean rank		6.42	7	7.58	2.67	3.33	4.33	3.33	1.33
Final rank		6	7	8	2	3	5	3	1



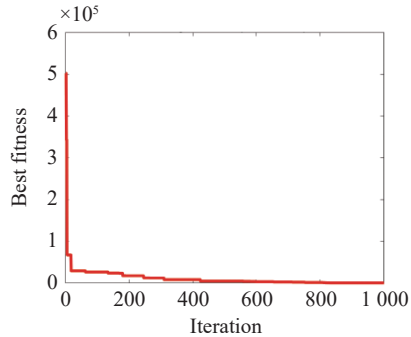
(a) 3-D map for 2-D function (CEC-2022-1)



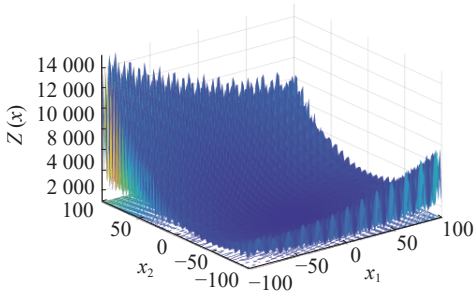
(b) Search history (CEC-2022-1)



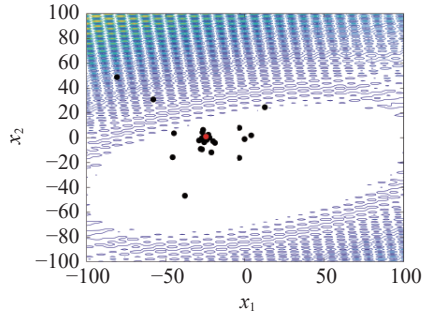
(c) Population diversity analysis (CEC-2022-1)



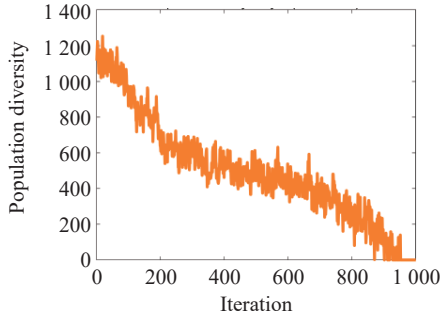
(d) Convergence curve (CEC-2022-1)



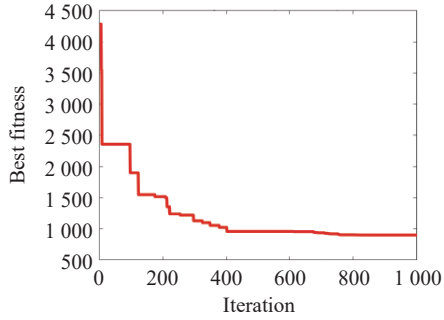
(e) 3-D map for 2-D function (CEC-2022-5)



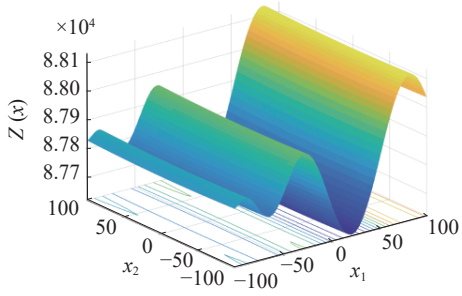
(f) Search history (CEC-2022-5)



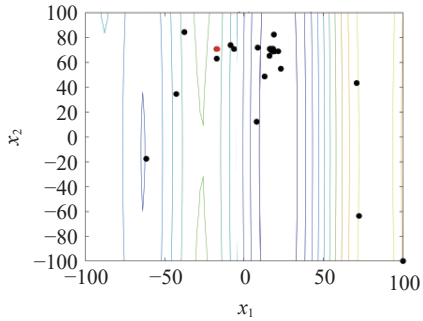
(g) Population diversity analysis (CEC-2022-5)



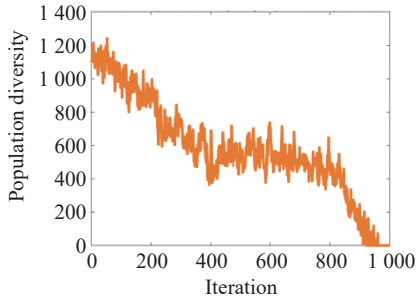
(h) Convergence curve (CEC-2022-5)



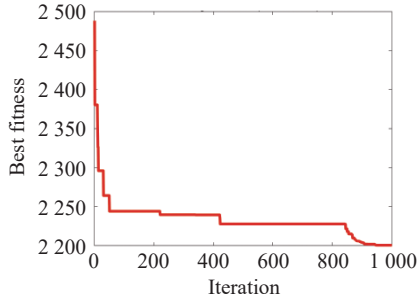
(i) 3-D map for 2-D function (CEC-2022-8)



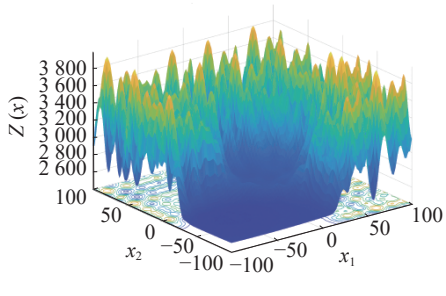
(j) Search history (CEC-2022-8)



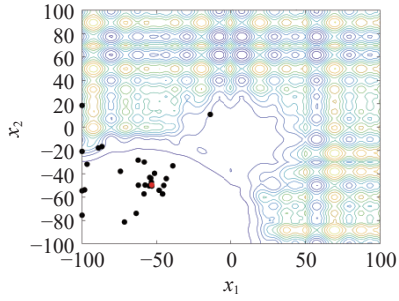
(k) Population diversity analysis (CEC-2022-8)



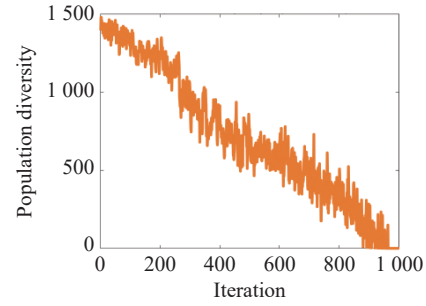
(l) Convergence curve (CEC-2022-8)



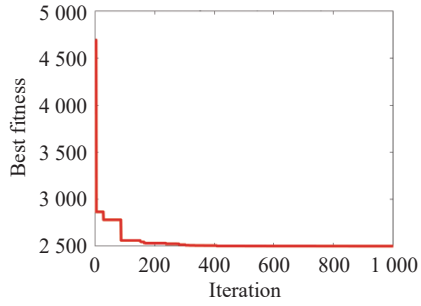
(m) 3-D map for 2-D function (CEC-2022-10)



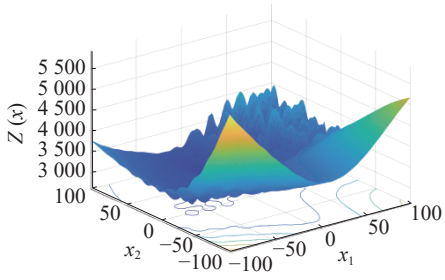
(n) Search history (CEC-2022-10)



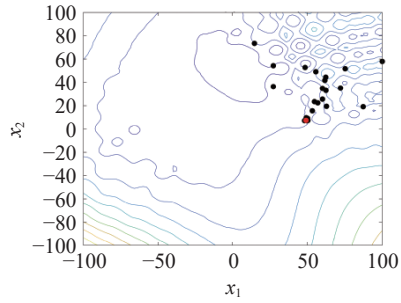
(o) Population diversity analysis (CEC-2022-10)



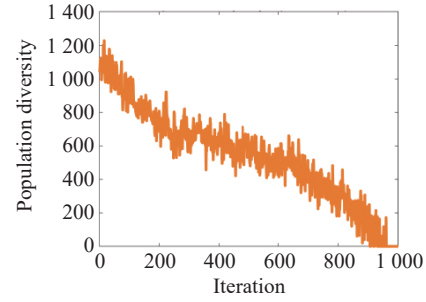
(p) Convergence curve (CEC-2022-10)



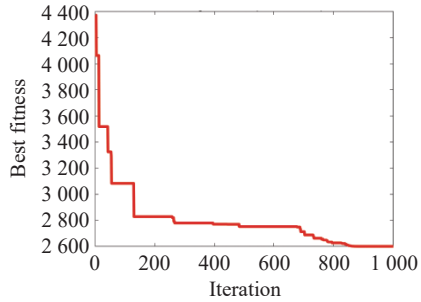
(q) 3-D map for 2-D function (CEC-2022-11)



(r) Search history (CEC-2022-11)

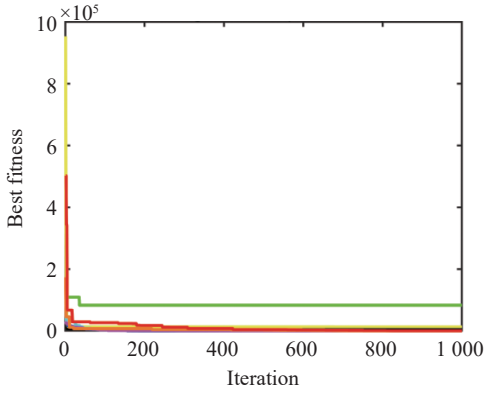


(s) Population diversity analysis (CEC-2022-11)

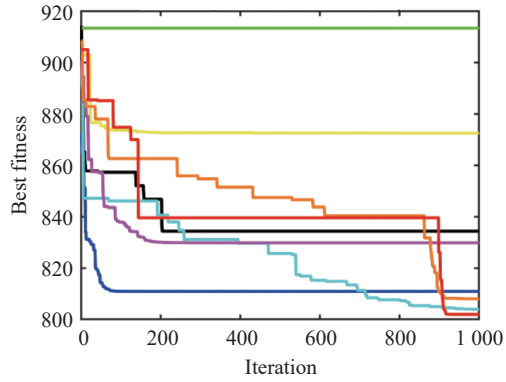


(t) Convergence curve (CEC-2022-11)

Fig. 1 Qualitative results of selected five benchmark test functions



(a) Convergence curve (CEC-2022-1)



(b) Convergence curve (CEC-2022-4)

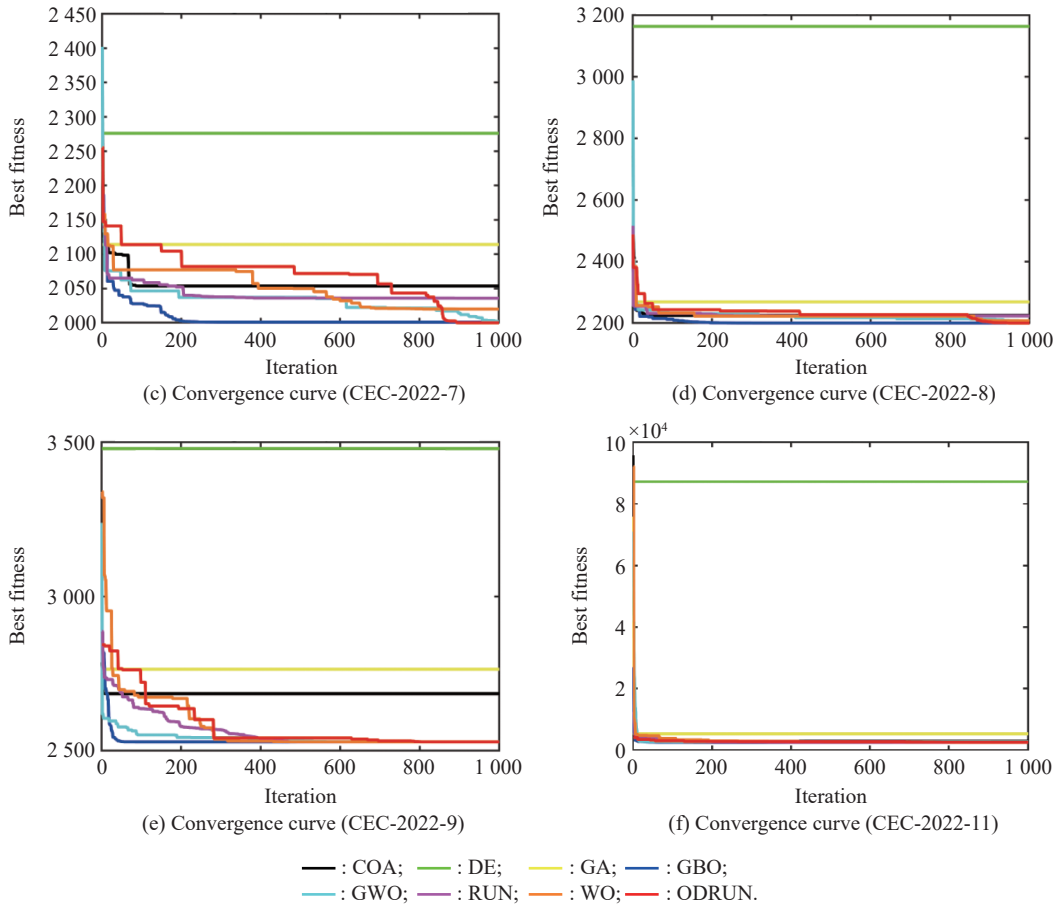
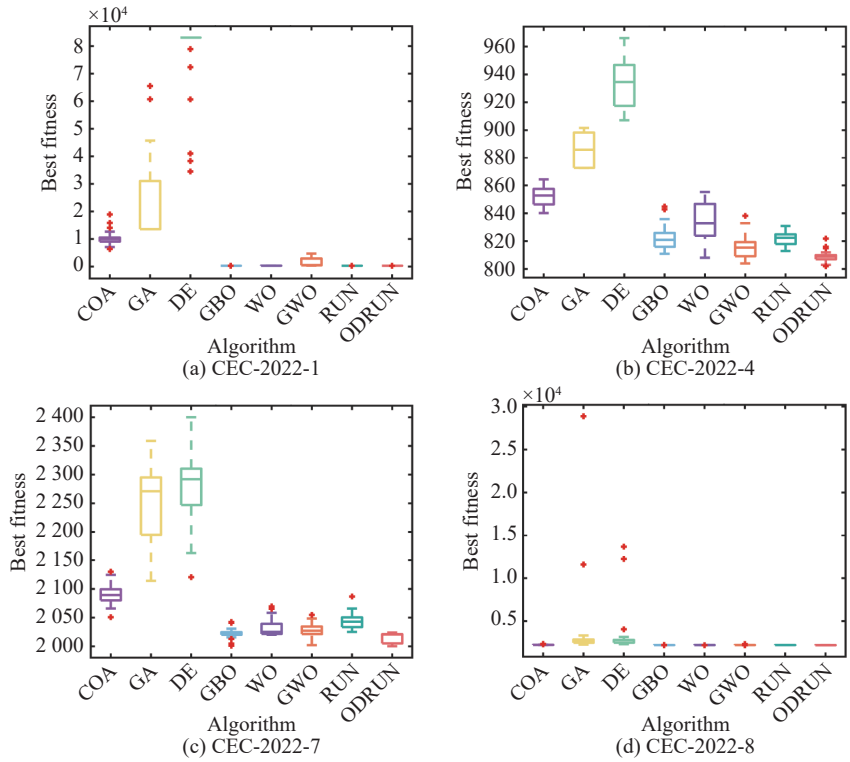
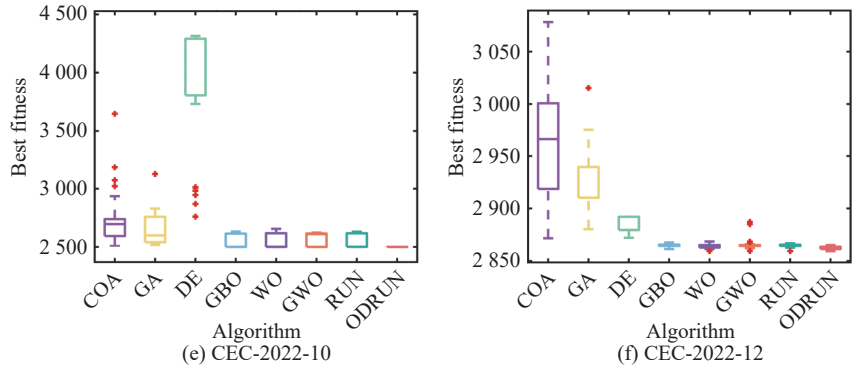


Fig. 2 Comparisons of convergence curves for six selected CEC2022 test functions





(e) CEC-2022-10 (f) CEC-2022-12

Fig. 3 Comparisons of boxplot graphs for six selected CEC2022 test functions

3.2 Qualitative results and analysis

The qualitative performance of ODRUN is evaluated using several functions. Five typical functions are selected from 12 benchmark functions for analysis. As shown in Fig. 1, the search history indicates that ODRUN follows a similar pattern to solve different problems. Initially, the algorithm increases exploration to discover promising regions in the search space and then shifts towards exploiting the best solutions. Regarding population diversity, there is a rich population diversity throughout the iterative process. Based on the convergence curves, we can observe a trend of gradual decline with some fluctuation.

3.3 Quantitative results and analysis

Table 2 reports the results of the seven algorithms on 10-dimensional problems, including best mean, standard deviation, and Friedman average ranks. Out of 36 cases, ODRUN achieved the best performance in 24 cases (66.67%), with its best, mean and standard deviation outperforming those of GBO (25%), and RUN (8.34%). The last two rows of tables display statistical analysis. The first row shows the Friedman mean rank, and the second row displays the final rank.

(i) Assessment of exploitative behavior. For unimodal functions (C_1^{22}), ODRUN shows strong exploitation abilities. This function has only one global best solution which is used to evaluate the local search capacity of optimization algorithms.

The ODRUN achieves the second-best optimal value in terms of best, mean, and standard deviation compared to other algorithms. While GBO ranks first, the performance of ODRUN, RUN, and GBO is very similar, with all three algorithms finding values very close to the optimal value of 300. The standard deviation for GBO, RUN, and ODRUN are 1.92E−08, 3.00E−03, and 6.63E−08, respectively. The minimal differences between these algorithms indicate that ODRUN, RUN, and GBO perform very similarly, with ODRUN exhibiting competitive exploitation abilities.

(ii) Assessment of exploration behavior. The multimodal functions (C_2^{22}−C_5^{22}) which have many local optimal solutions. The ODRUN algorithm achieved the best maximum values, average values, and standard deviations across 9 out of 12 performance evaluation metrics, accounting for 75% of the total. This outperforms the GBO algorithm (16.67%), the RUN algorithm (8.34%), and other comparison algorithms (0%), demonstrating its superior global optimization capabilities.

(iii) Ability to avoid local optima. Hybrid functions (C_6^{22}−C_8^{22}) are used to evaluate the ability to avoid local optima, as they are highly complex. ODRUN performed well, with 5 out of 9 cases (55.56%), achieving the best, mean and standard deviation which is higher than GBO (33.34%) and RUN (22.22%), while other algorithms get trapped in local optima, ODRUN continues to explore better solutions.

(iv) Comprehensive performance evaluation. Composition functions (C_9^{22}−C_{12}^{22}) combine multiple sub-functions to create complex optimization problems to increase the complexity of optimization problems helping to assess both their robustness and overall performance. The ODRUN outperforms the comparison algorithms with 10 out of 12 cases (83.37%) achieving better performance in terms of the best, mean and standard deviation compared to GBO (16.67%) and other algorithms (0%) in multimodal functions.

(v) Convergence ability. The convergence curves of COA, GA, DE, WO, GWO, RUN, and ODRUN are depicted in Fig. 2. The ODRUN algorithm has higher convergence accuracy than other algorithms which can achieve lower objective function values. For example, on functions C_4^{22}, C_7^{22}, and C_{10}^{22}, the ODRUN algorithm demonstrates higher convergence accuracy. ODRUN achieves fast convergence on specific test functions such as C_1^{22}, C_8^{22}, C_{12}^{22}. While GWO and RUN converge faster initially, their solutions are less accurate, which can be

demonstrated by combining the results in Table 2.

3.4 Stability and robustness analysis

From the boxplots shown in Fig. 3, it can be noted that the distribution of the best-fitness values obtained by these eight algorithms over 30 independent runs shows distinct profiles. Overall, the boxplot for the ODRUN algorithm appears to be narrow in the majority of the test functions with no significant outlier, such as C_7^{22}, C_8^{22}, C_9^{22}, C_{10}^{22}, C_{12}^{22}.

This indicates that the data derived from the ODRUN algorithm shows higher stability and more stable performance compared to the other algorithms. The results show that the stability of most algorithms, such as COA, GA, and DE, is obviously weakened, and COA in C_{12}^{22} is one of the obvious examples. The Wilcoxon-signed test is applied as shown in Table 3. Hereof, "Sig" reveals the significance of a hypothesis test with a significance level of 0.05. "H = 1" reveals the ODRUN has a significant difference from the comparison algorithm for the test function, and "H = 0" reveals the ODRUN has no significant difference from the comparison algorithm for the test function. Most of the values are less than 0.05, which shows that ODRUN significantly improves performance.

Table 3 Wilcoxon signed test for CEC2022 benchmark functions

Function	Statistics	COA	GA	DE	GBO	WO	RUN
C_1^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	2.84E−05	1.73E−06	1.73E−06
	H	1	1	1	1	1	1
C_2^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	0.003 0	4.29E−06	0.000 1
	H	1	1	1	1	1	1
C_3^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	0.011 7	1.49E−05	1.13E−05
	H	1	1	1	1	1	1
C_4^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	2.35E−06	0.000 8	2.88E−06
	H	1	1	1	1	1	1
C_5^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	2.60E−06	0.000 1	4.73E−06
	H	1	1	1	1	1	1
C_6^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	0.125 4	3.72E−05	0.749 9
	H	1	1	1	0	1	0
C_7^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	0.008 7	0.001 2	6.34E−06
	H	1	1	1	1	1	1
C_8^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	0.003 9	5.79E−05	1.13E−05
	H	1	1	1	1	1	1
C_9^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	3.11E−05	1.73E−06	1.73E−06
	H	1	1	1	1	1	1
C_{10}^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	3.18E−06	2.16E−05	1.73E−06
	H	1	1	1	1	1	1
C_{11}^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	0.008 7	1.73E−06	1.73E−06
	H	1	1	1	1	1	1
C_{12}^{22}	Sig.	1.73E−06	1.73E−06	1.73E−06	5.79E−05	0.000 4	6.04E−03
	H	1	1	1	1	1	1

3.5 Rank analysis

Table 4 shows the Friedman mean ranks and average rank for the mean fitness values, best fitness values, standard deviation from ODRUN and seven other algorithms on CEC2022 test functions. ODRUN ranks first in 24 cases, where these 24 cases consist of three performance metrics (best, mean, and standard deviation). These results indicate that the ODRUN algorithm performs the best among the eight algorithms on test functions.

Table 4 Ranks for the mean, best, standard deviation for CEC2022 benchmark functions

Algorithm	Mean value rank													
	C_1^{22}	C_2^{22}	C_3^{22}	C_4^{22}	C_5^{22}	C_6^{22}	C_7^{22}	C_8^{22}	C_9^{22}	C_{10}^{22}	C_{11}^{22}	C_{12}^{22}	Average rank	Final rank
COA	6	8	6	6	6	6	6	6	6	7	6	8	6.42	6
GA	7	7	7	7	7	7	7	8	7	6	7	7	7.00	7
DE	8	6	8	8	8	8	8	7	8	8	8	6	7.58	8
GBO	1	2	3	4	3	1	2	2	3	2	4	4	2.58	2
WO	4	5	2	5	2	4	4	4	2	4	2	2	3.33	3
GWO	5	4	4	2	4	5	3	5	5	5	5	5	4.33	5
RUN	3	1	5	3	5	3	5	3	4	3	3	3	3.42	4
ODRUN	2	3	1	1	1	2	1	1	1	1	1	1	1.33	1
Algorithm	Best value rank													
	C_1^{22}	C_2^{22}	C_3^{22}	C_4^{22}	C_5^{22}	C_6^{22}	C_7^{22}	C_8^{22}	C_9^{22}	C_{10}^{22}	C_{11}^{22}	C_{12}^{22}	Average rank	Final rank
COA	6	6	6	6	6	6	6	6	6	7	6	8	6.25	6
GA	7	7	7	7	7	8	7	7	8	6	7	7	7.08	7
DE	8	8	8	8	8	7	8	8	7	8	8	6	7.67	8
GBO	1	1	1	4	4	1	2	2	1	3	1	5	2.17	2
WO	4	4	3	3	2	3	4	4	3	4	4	3	3.42	3
GWO	5	5	4	2	3	5	3	3	5	2	5	4	3.83	4
RUN	3	2	5	5	5	4	5	5	4	5	3	2	4.00	5
ODRUN	2	3	2	1	1	2	1	1	2	1	2	1	1.58	1
Algorithm	Standard value rank													
	C_1^{22}	C_2^{22}	C_3^{22}	C_4^{22}	C_5^{22}	C_6^{22}	C_7^{22}	C_8^{22}	C_9^{22}	C_{10}^{22}	C_{11}^{22}	C_{12}^{22}	Average rank	Final rank
COA	6	8	8	5	6	6	6	6	6	7	7	8	6.58	7
GA	8	7	5	6	7	7	7	8	7	6	2	7	6.42	6
DE	7	6	7	8	8	8	8	7	8	8	8	6	7.42	8
GBO	1	2	4	4	3	3	2	2	3	3	5	4	3.00	2
WO	4	5	2	7	2	4	5	3	2	5	6	3	4.00	5
GWO	5	4	3	3	4	5	3	5	5	2	3	5	3.92	4
RUN	3	3	6	2	5	1	4	1	4	4	4	2	3.25	3
ODRUN	2	1	1	1	1	2	1	4	1	1	1	1	1.42	1

3.6 Mechanism comparison experiment

In the previous section, an odd-even dimension and neighbor search strategies are developed to improve the performance of RUN. To validate the effectiveness of these mechanisms, five different variants of RUN are evaluated on the 12 benchmark functions from CEC2022. The details of these variants are provided in Table 5, where "Yes" indicates that a given strategy is incorporated into the RUN variant, while "No" indicates that the strategy is not implemented. Each algorithm is tested 30 times with 1 000 iterations and a population size of 30. Table 6 shows the detailed ranking of algorithms. ODRUN ranks first in all metrics, demonstrating that its mechanism fusion is the optimal choice.

Table 5 Five different variants of RUN

Variant	Enhanced solution quality scheme	Odd-even dimension strategy	Neighbor search strategy
RUN	Yes	No	No
RUNOD	Yes	No	Yes
RUNNS	Yes	Yes	No
RUNDS	Yes	Yes	Yes
ODRUN	No	Yes	Yes

Table 6 Performance ranking of variants

Variant	Average rank	Final rank
RUN	4.28	5
RUNOD	3.72	4
RUNNS	3.06	3
RUNDS	2.11	2
ODRUN	1.84	1

3.7 Sensitivity analysis of ODRUN

The sensitivity analysis of the control parameters (i.e., a and b) indicates that the performance of ODRUN is insensitive to parameter changes. To explore different control parameter combinations, the algorithm was tested on 12 benchmark functions from the CEC2022. The values of the control parameters were varied as follows: $a \in \{5,10,20,30,40\}$ and $b \in \{4,8,12,16,20\}$. This resulted in 25 distinct combinations of the two parameters. Each combination was evaluated by the mean values obtained from 30 independent runs. The mean rank values for each combination on the benchmark functions are presented in Fig. 4.

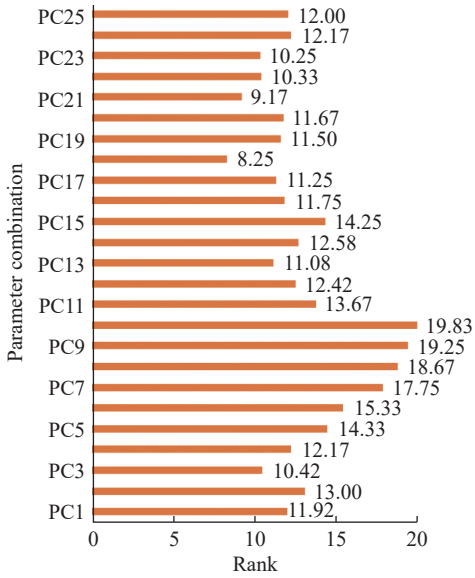


Fig. 4 Average rank of different parameter combinations

The results show that the optimal parameter combination is associated with PC13 (with $a = 20$ and $b = 16$), and the performance is nearly equivalent to that of PC18 (with $a = 5$ and $b = 20$). Additionally, the rankings of most parameter combinations are tightly clustered, suggesting that the algorithm's performance is not significantly influenced by variations in the control parameters.

4. Experimental 2: applying ODRUN for solving IJRPTC

To further assess the proposed algorithm, the improved ODRUN algorithm is employed to solve this proposed problem which is a discrete mixed-integer nonlinear nondeterministic polynomial hard problem. The complexity of the problem stems from its high nonlinearity and the intricate interactions between decision variables, making it challenging for traditional optimization methods.

4.1 IJRPTC: a typical NP-hard problem

The joint replenishment problem (JRP) is a well-known optimization problem in inventory management, grouping items into the same order from a supplier can reduce the total cost [22–24]. Both JRP and its extension have been proven to be the NP-hard problem [25,26]. Numerous researchers are devoted to finding effective and efficient methodologies for solving these problems [27–29]. They ignore the widespread application of trade credit in supply chain management. However, trade credit plays a pivotal role in global trade activities [30,31]. In commercial practice, suppliers usually provide a fixed trade credit period time and allow retailers to delay payment. During this period, retailers do not have to pay interest and can earn interest from the accumulated sales, but if the retailer fails to pay the full amount at the end of the credit period, they need to pay the capital occupancy cost for the products in stock. In addition, controlling the flow of materials from suppliers to end customers is a challenge. Due to various factors, such as imperfect transportation processes, production batch instability, and uncertainties in the purchasing process lead to a percentage of imperfect items in procurement batches, which affects the cost-saving of joint purchasing policy and increases the capital cost.

The simple description of the novel problem is illustrated in Fig. 5.

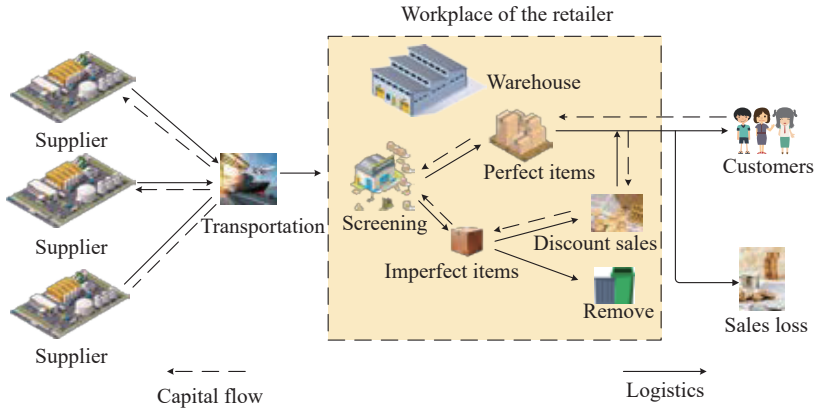


Fig. 5 Simple description of the proposed IJRPTC

It considers a supply chain consisting of a single retailer and multiple suppliers, where the retailer jointly replenishes multiple items. In each purchase cycle, suppliers offer trade credit policies, allowing the retailer to delay payment of the purchase amount for a permitted length of credit period. Once the order arrives at the retailer's location, the screening operation begins. Given the extent of imperfect items may be different, a mixed strategy is applied in which imperfect items with less damage are sold at a discounted price to customers, while unusable imperfect items are scrapped and removed from the retailer's inventory once the screening time is over. Fig. 6 shows the inventory level of retailers under the proposed problem.

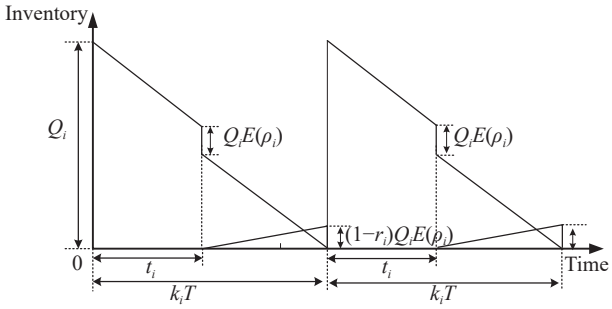


Fig. 6 Inventory level of the retailer under the proposed problem

By referring to the work of [27] and [32], the following assumptions are made in constructing the JRP model with defective items and trade credit:

(i) The demand rate D_i for each item is known and remains constant.

(ii) The items are independent and replenished from a single supplier.

(iii) No shortages are allowed.

(iv) The screening operation begins as soon as the order arrives at the retailer's location.

(v) Denote ρ_i is the percentage of imperfect for item i in each replenishment, and the purchased quantity for each item is Q_i. The expected number of perfect units is $Q_i(1-\rho_i)$ and the expected number of imperfect items is $Q_i\rho_i$.

(vi) Each item is screened at a rate γ_i ($\gamma_i > D_i$). Hence the screening time t_i to finish the order quantity of item i is $t_i = Q_i/\gamma_i$. Once the screening is over, the imperfect items are marked down.

(vii) Under the indirect grouping strategy, the replenishment cycle for each item is $T_i = k_iT$, where k_i is an integer multiple of T. During the period k_iT, the expected number of perfect items must meet the customer demand, hence, $Q_i(1-\rho_i) = k_iTD_i$ and $Q_i(1-\rho_i) \geqslant D_it_i$ must satisfy. Additionally, the number of perfect items is at least equal to the demand, $\gamma_i(1-\rho_i) \geqslant D_i$ during the screening time t_i.

(viii) Denote r_i is a fractional percentage of unusable imperfect items. Imperfect items $(1-r_i)Q_i\rho_i$ with less damage are sold to customers at discount price $(1-\theta_i)p_i$, while unusable imperfect products $r_iQ_i\rho_i$ are scrapped and removed from the retailer's inventory with a loss cost δ_i per unit.

(ix) Within the trade credit period M, the account remains unsettled, and accumulated sales revenue is assumed to be deposited into an account with an interest rate of I_e. When the permissible delayed time is over, and retailers pay for the inventory items with an interest rate of I_p.

The notations used are presented in Table 7. The total cost of IJRPTC is composed of five parts, including the holding cost, ordering cost, screening total cost, loss cost, interested charged and earned.

Table 7 Notations description

Parameter	Description
n	Number of items
D_i	Demand rate for item i
Q_i	The order quantity for item i
S	Major ordering costs associated with each replenishment
s_i	Minor ordering cost per order of item i
h_i	Holding cost per unit of item i at the retailer
ρ_i	The percentage of imperfect products of item i in each replenishment cycle
γ_i	The screening rate for item i, $\gamma_i > D_i$
α_i	The screening cost per unit time for item i
M	The length of the trade credit period
p_i	Retail price per unit for the item i
c_i	Purchasing unit price for the item i
I_p	Interest charged per dollar by the supplier
I_e	Interest earned per dollar for the retailer $I_e \leqslant I_p$
θ_i	The discounted rate for the proportion of imperfect items
r_i	The percentage of unusable imperfect items
δ_i	The loss per unit of unusable imperfect items for item i
t_i	The screening time for item i
T	Basic cycle time (decision variable)
k_i	The integer number that decides the replenishment cycle time of item i (decision variable)

These costs are as follows:

(i) Total annual stock holding cost

$$C_h = \sum_{i=1}^{n}\left[\frac{k_iTD_ih_i}{2} + \frac{h_ik_iTD_i^2\rho_i}{\gamma_i(1-\rho_i)^2} + \frac{Q_i\rho_i(1-r_i)h_i}{2}\right] \quad (22)$$

(ii) Total ordering cost

$$C_o = \frac{A}{T} + \sum_{i=1}^{n} \frac{a_i}{k_i T} \tag{23}$$

(iii) Total screening cost per cycle

$$C_s = \sum_{i=1}^{n} \frac{\alpha_i D_i}{(1-\rho_i)} \tag{24}$$

(iv) Loss cost of recalling the defective items

$$C_l = \sum_{i=1}^{n} \frac{\delta_i r_i D_i \rho_i}{(1-\rho_i)} \tag{25}$$

(v) Interested charged and earned

During the credit period, the retailer incurs two types of costs: the interest earned from sales revenue and the interest charged for unsold inventory beyond the trade credit period. These costs depend on three scenarios defined by the relationships between M, t_i, and k_iT. Items can be categorized into three sets based on their time intervals: J_1 represents the set of items satisfying $M \leqslant t_i \leqslant k_iT$; J_2 represents the set of items satisfying $t_i \leqslant M \leqslant k_iT$ and J_3 represents the set of items satisfying $k_iT \leqslant M \leqslant k_iT + t_i$. Hence, the interest earned and interest paid are given by

$$\begin{aligned} C_{\mathrm{IE}} = & \sum_{i\in J_1} \frac{p_i I_e D_i M^2}{2k_iT} + \sum_{i\in J_2} \frac{p_i I_e D_i (k_iT)^2}{8k_iT} + \\ & \sum_{i\in J_2} \frac{p_i I_e (2M + k_iT)(2M - k_iT) D_i}{8k_iT} + \\ & \sum_{i\in J_2} \frac{(1-\theta_i) p_i I_e (M - t_i)(1 - r_i) Q_i \rho_i}{k_iT} + \\ & \sum_{i\in J_3} \frac{D_i p_i I_e (k_iT)^2}{2k_iT} + \sum_{i\in J_3} \frac{p_i I_e (\mathrm{M}\text{-}k_iT) k_iT D_i}{k_iT} + \\ & \sum_{i\in J_3} \frac{(1-\theta_i) p_i I_e (k_iT - t_i)(1 - r_i) Q_i \rho_i}{k_iT}, \end{aligned} \tag{26}$$

$$\begin{aligned} C_{\mathrm{IP}} = & \sum_{i\in J_1} \frac{c_i I_P D_i (k_iT - M)^2}{2k_iT} + \sum_{i\in J_1} \frac{c_i I_P (M - k_iT) D_i \rho_i}{1-\rho_i} + \\ & \sum_{i\in J_2} \frac{c_i I_P D_i (k_iT - M)^2}{2k_iT}. \end{aligned} \tag{27}$$

The objective of IJRPTC is to minimize the total cost by determining the optimal replenishment frequency:

$$\begin{aligned} & \text{Minimize} TC(\boldsymbol{k}, T) = C_h + C_o + C_s + C_l + C_{\mathrm{IP}} - C_{\mathrm{IE}} \\ & \text{s.t. } k_i \in \mathbf{Z}^+, \ \forall i \in n \\ & T \in (0,1]. \end{aligned} \tag{28}$$

The integrality condition of the replenishment schedule time is the constraint that defines the domain of the basic cycle time.

4.2 ODRUN-based procedures for IJRPTC

The proposed ODRUN is designed to address the proposed discrete mixed-integer nonlinear model. The objective of the model is to determine the optimal replenishment frequency of each item to minimize the total cost. The replenishment frequency of each items $\boldsymbol{kT} = [k_1T, k_2T, \cdots, k_nT]$ is determined by the discrete integer variables $\boldsymbol{k}$ and continuous variable T. To address IJRPTC, an encoding and decoding scheme is employed. The solution generated by ODRUN is $\boldsymbol{x} = (k_1, k_2, \cdots, k_n; T)$, where the dimension of this specific problem is $n+1$, the n elements indicate the replenishment frequency for each item, and the final element represents the basic replenishment cycle time for all items. Initially, each population's values are randomly generated within the search space range of (0,1), and then each solution is decoded into a feasible solution. The decoding procedure can be formulated as

$$k_i = \mathrm{floor}(k\mathrm{code}_i (k_{\mathrm{ub}} - k_{\mathrm{lb}}) + k_{\mathrm{lb}}), \tag{29}$$

$$T = \mathrm{round}(T\mathrm{code} (T_{\mathrm{ub}} - T_{\mathrm{lb}}) + T_{\mathrm{lb}}), \tag{30}$$

where the lower bound k_{lb} and upper bound k_{ub} of k_i are generally set as 1 and 10 [45]. Additionally, the unit T, representing years, generally lies in (0.0001,1) in this paper. Supposing a problem of six items ($n=6$), the solution generated by ODRUN is $\boldsymbol{x} = [0.012\,7, 0.004\,8, 0.027\,8, 0.012\,6, 0.148\,4, 0.275\,4, 0.107\,2]$, by employing (29) and (30), the solution is decoded to $\boldsymbol{x} = [1,1,1,1,2,3,0.1072]$, the replenishment frequency of each item is $\boldsymbol{kT} = [0.107\,2, 0.107\,2, 0.107\,2, 0.107\,2, 0.214\,4, 0.321\,6]$. Fig. 7 shows the structure of a solution and the decoding transformation.

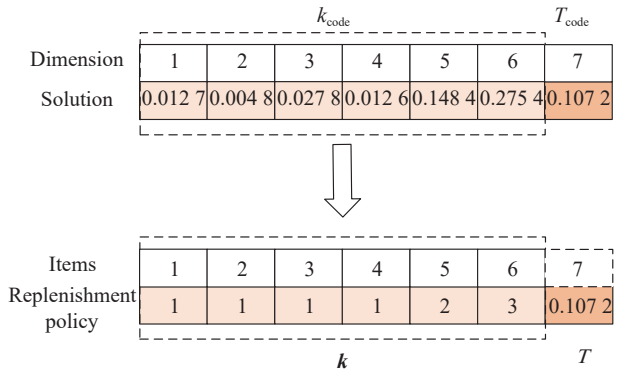


Fig. 7 Structure of a solution and decoding transformation (n=6)

The procedure of ODRUN for the proposed JRP with imperfect items and trade credit is detailed as follows:

Step 1 Initialization. Set parameters for the ODRUN, including the maximum number of iterations and population size.

Step 2 Calculate the objective value. When evaluating the objective function, $\boldsymbol{k}$ and $\boldsymbol{T}$ are decoded into their practical values using (29) and (30). Calculate the total cost using (28).

Step 3 Update the positions based on ODRUN procedures. Firstly, generate a new position. X_{new} based on the odd-even dimension search strategy RK4 mechanism using (15). Furthermore, perform the modified enhanced solution quality mechanism based on the neighbor learning searching strategy using (21).

Step 4 Compare the total cost and select the best solution. Repeat until the current iteration times reaches the maximum iteration.

4.3 Performance evaluation on large-scale IJRPTC

To assess the performance of ODRUN, large-scale randomized problems are generated. The problem scale n is uniformly distributed from 3 to 80, and 10 groups are randomly generated. The major ordering cost S is set to 150 or 300, resulting in 20 groups. Each group contained 30 randomly generated problem instances, yielding a total of 600 instances (30 problem examples per group × 20 groups) for testing. The parameter ranges for the problem examples are shown in Table 8, as referenced in [21] and [32]. The parameter settings of algorithms are shown in Table 1. The population size is established at 50, and the maximum iteration is fixed at 200. All algorithms are run 10 times to obtain convincible results. The mean values for the average total cost, best total cost, and standard error across each group of 30 problem examples were computed, results can be found in Table 9 and Table 10. The maximum improvement ratio (MIR), which compares the ODRUN algorithm's results to the best results obtained by any other algorithm, is presented in the last column of these tables.

Table 8 Ranges of the parameters

Parameter	Range
n	[3,80]
D_i	[1 000, 5 000]
S	{150,300}
s_i	[35,55]
h_i	[0.5,2.5]
ρ_i	[0.01,0.2]
γ_i	$[1.25\mathrm{D}_i, 2\mathrm{D}_i]$
M	[10/365,30/365]
p_i	[40,60]
c_i	[2,4]
α_i	$[0.01c_i, 0.05c_i]$
I_e	[0.06,0.12]
I_p	$[1.2I_e, 2I_e]$
r_i	[0, 0.2]
θ_i	[0.05, 0.5]
δ_i	$[0.1c_i, 0.2c_i]$

Table 9 Comparative results of proposed algorithms under various scale problems (S=150)

n	Indicator	COA	GA	DE	GBO	WO	GWO	RUN	ODRUN	MIR/%
14	Best	13 360.82	13 280.98	13 344.55	15 589.10	13 280.81	13 280.98	13 287.55	13 280.81	14.81
	Mean	13 873.80	13 306.69	13 408.01	15 683.90	13 284.57	13 300.96	13 336.34	13 280.87	15.32
	Standrad	471.01	35.07	41.21	110.94	6.92	25.24	46.91	0.08	99.98
22	Best	20 227.18	20 399.64	19 619.05	23 036.99	19 419.36	19 423.16	19 473.84	19 417.17	15.71
	Mean	21 218.65	20 681.84	19 724.16	23 272.98	19 436.99	19 483.51	19 606.78	19 418.14	16.56
	Standrad	576.35	154.84	60.78	278.63	24.14	82.48	95.81	1.85	99.68
29	Best	26 286.61	26 554.24	25 208.75	29 874.27	24 863.14	24 865.78	25 008.02	24 851.06	16.81
	Mean	27 214.02	26 633.08	25 338.24	30 176.36	24 909.13	24 999.15	25 182.16	24 859.55	17.62
	Standrad	727.40	49.61	71.06	355.58	45.45	175.17	112.55	12.69	98.26
34	Best	31 384.04	31 487.49	30 005.48	35 156.72	29 531.36	29 542.37	29 736.63	29 512.66	16.05
	Mean	32 327.33	31 668.04	30 144.23	35 477.32	29 612.97	29 755.27	29 958.42	29 516.84	16.80
	Standrad	762.86	115.26	70.89	304.03	67.09	330.52	139.35	6.68	99.12
47	Best	41 792.41	42 412.31	40 084.12	47 206.52	39 203.00	39 214.49	39 691.70	39 128.89	17.11
	Mean	42 800.57	42 754.88	40 257.54	47 903.28	39 358.40	39 568.41	39 992.77	39 168.49	18.23
	Standrad	780.02	200.17	88.30	951.98	118.10	445.39	188.88	40.89	95.71
50	Best	45 735.97	46 467.23	43 674.30	51 908.69	42 799.37	42 805.35	43 299.32	42 745.50	17.65
	Mean	46 727.50	46 822.95	43 837.39	52 472.18	42 965.72	43 171.54	43 586.39	42 758.43	18.51
	Standrad	693.75	222.05	79.07	887.75	126.15	462.79	182.13	20.40	97.70

Continued

n	Indicator	COA	GA	DE	GBO	WO	GWO	RUN	ODRUN	MIR/%
	Best	47935.40	48769.61	45844.50	54336.15	44874.15	44869.02	45400.21	44788.27	17.57
53	Mean	48988.86	49067.57	46004.07	55440.64	45043.92	45288.95	45718.49	44845.32	19.11
	Standrad	793.43	180.57	83.79	1632.03	130.83	547.71	191.04	56.71	96.52
	Average	54525.74	55349.47	51975.04	61727.35	50841.37	50825.43	51528.87	50699.26	17.87
60	Standrad	55458.57	55705.93	52126.53	62603.58	51042.73	51272.12	51863.33	50729.79	18.97
	Rank	698.79	221.37	82.85	1456.31	141.92	589.18	202.46	32.80	97.75
	Average	65187.62	65953.77	61657.99	73773.35	60421.13	60319.52	61207.98	60166.58	18.44
73	Standrad	65969.64	66447.32	61849.01	74847.00	60673.89	60839.22	61592.65	60297.17	19.44
	Rank	591.92	293.44	92.97	1371.03	181.29	701.74	222.35	106.59	92.23
	Average	71996.13	72772.03	68383.16	80855.50	66872.49	66721.60	67845.62	66477.94	17.78
80	Standrad	72771.34	73316.83	68539.46	82205.43	67167.15	67208.77	68228.04	66566.55	19.02
	Rank	570.60	323.06	85.55	2354.88	195.96	549.47	235.44	68.91	97.07

Table 10 Comparative results of proposed algorithms under various scale problems (S=300)

n	Indicator	COA	GA	DE	GBO	WO	GWO	RUN	ODRUN	MIR/%
	Best	13176.05	13173.18	13172.52	15009.06	13172.52	13172.62	13173.35	13172.52	12.24
12	Mean	13198.99	13204.36	13172.52	15027.95	13176.27	13173.10	13184.35	13172.60	12.35
	Standard	18.34	71.75	0.00	21.13	7.09	1.41	15.84	0.04	99.94
	Best	14269.00	14263.68	14263.09	16480.57	14263.09	14263.09	14263.94	14263.09	13.46
14	Mean	14289.40	14314.37	14263.53	16499.80	14267.53	14264.59	14274.25	14263.27	13.55
	Standard	18.23	90.25	0.75	20.36	7.40	3.35	13.21	0.20	99.78
	Best	18025.59	17989.39	17961.06	20889.83	17961.21	17961.12	17970.77	17961.06	14.02
18	Mean	18074.02	18436.98	17961.81	20964.81	17983.46	17966.98	18026.19	17961.27	14.33
	Standard	31.65	528.98	1.36	57.75	28.06	10.55	43.53	0.52	99.90
	Best	19979.44	20021.93	19930.90	23120.97	19928.45	19927.06	19937.75	19926.95	13.81
20	Mean	20028.80	20701.39	19939.80	23199.79	19946.17	19931.85	19986.59	19927.15	14.11
	Standard	36.78	629.75	7.84	60.01	26.10	8.96	39.78	0.08	99.99
	Best	30441.85	32473.26	32183.16	35725.52	30086.91	30075.03	30241.39	30066.21	15.84
33	Mean	30536.76	33696.09	32709.69	36022.17	30187.02	30127.89	30406.70	30078.00	16.50
	Standard	54.41	720.17	281.69	389.33	80.28	49.39	115.56	17.32	97.59
	Best	44144.82	47636.31	47799.17	51517.69	43371.83	43340.73	43803.39	43272.13	16.01
49	Mean	44254.57	49017.05	48167.00	51857.90	43561.05	43482.69	44068.71	43314.49	16.47
	Standard	60.20	1088.73	236.74	348.67	130.70	108.10	159.48	45.85	95.79
	Best	48622.66	52581.08	52932.77	56924.28	47734.65	47723.45	48324.95	47616.21	16.35
54	Mean	48757.41	53989.87	53349.79	57423.08	47953.15	47896.60	48588.23	47642.40	17.03
	Standard	71.96	1134.34	264.34	696.64	154.16	132.27	157.59	31.01	97.27
	Best	49335.94	53374.58	53692.51	57958.94	48418.33	48390.12	48932.75	48249.62	16.75
56	Mean	49465.47	54789.61	54168.12	58324.35	48633.23	48561.93	49252.60	48284.09	17.21
	Standard	72.18	1122.85	302.91	426.30	152.58	127.72	189.03	38.48	96.57
	Average	60809.67	66236.15	67423.51	71777.09	59699.85	59649.60	60404.50	59476.88	17.14
69	Standard	60975.58	67790.59	67962.74	72544.73	59955.93	59880.43	60751.30	59582.37	17.87
	Rank	85.63	1231.32	353.65	1346.98	172.69	159.40	191.34	88.64	93.42
	Average	69256.58	75356.31	76772.37	81922.72	67820.88	67858.41	68831.98	67483.35	17.63
80	Standard	69431.03	76757.97	77349.30	82690.58	68161.41	68162.55	69201.00	67575.79	18.28
	Rank	87.79	994.13	356.35	1142.17	265.52	199.51	207.07	79.92	93.00

(i) Comparison of effectiveness

Table 9 and Table 10 show the average cost and best-found total cost from 10 independent runs of each algorithm in each group. Some conclusions can be drawn: i) ODRUN achieves the lowest average total cost in 19 out of 20 groups, with a success rate of 95%, followed by DE, which achieves the lowest in only 1 out of 20 groups (5% success rate). ii) ODRUN finds the lowest average best value in all groups, with a success rate of 100%. The next best performers are GWO, WO, and DE, each with success rates of 10%, 10%, and 5%, respectively. iii) The advantages of ODRUN in finding the optimal values are clearer. The MIR ranges from 14.81% to 19.5%, which indicates significant cost savings in average cost and best-found total cost.

(ii) Comparison of stability

Table 9 and Table 10 also display the average standard deviation of each algorithm in each group. The ODRUN shows superior stability compared to the other seven algorithms in solving this well-known NP-hard problem.

The key observations include: i) ODRUN achieves the smallest standard deviation in 17 out of 20 groups. The highest standard deviation observed is 106.59 when the number of items $n = 73$ and $S = 150$, and the smallest standard deviation observed is 0.04 when the number of items $n = 12$ and $S = 300$. ii) The MIR for standard deviation is 99.98%, which indicates the largest improvement ratio of ODRUN compared to other algorithms. iii) As the number of items increased, the stability advantage of the ODRUN algorithm remained unaffected. For example, in a case with 14 items, the standard deviation of ODRUN is 0.08, while the standard deviation of the GA and RUN algorithms are 35.07 and 46.91, respectively. In a case with 80 items, ODRUN has a standard deviation of 68.91, compared to 2354.88 for GBO and 235.44 for RUN.

(iii) Comparison of robustness

The algorithm's performance remains unaffected under different problem settings. As the number of items increases, ODRUN can always achieve the optimal average total cost in all cases.

In the large-scale randomly generated instances, ODRUN outperforms the other algorithms in 56 out of 60 cases, where these 60 cases consist of three performance metrics (best, mean, and standard deviation) for 20 groups, achieving an effectiveness rate of 93.34%. This demonstrates ODRUN's robust and superior overall performance.

5. Conclusion and further research

The paper proposes an ODRUN algorithm, which integrates an odd-even dimension search strategy with the fourth-order Runge-Kutta method and a modified enhanced solution quality scheme based on a neighbor search strategy. This algorithm exhibits excellent exploration and exploitation capabilities, convergence accuracy, stability, and robustness. The improved algorithm achieved the best performance in 24 out of 36 cases across three statistical indicators (best, mean, and standard deviation) for 12 benchmark functions, with an overall effectiveness ratio of 66.67%. It outperforms seven state-of-the-art metaheuristic intelligent algorithms, including COA, GA, DE, GBO, WO, GWO, and RUN. Ranking analysis indicates that the ODRUN algorithm outperforms the comparison algorithms, with first-place rankings across multiple statistical metrics. This performance is further validated by the Friedman mean ranks and Wilcoxon signed-rank test, which confirm its superiority over other algorithms. Furthermore, in the large-scale IJRPTC, a well-known, highly nonlinear, discrete mixed-integer NP-hard problem in inventory management, ODRUN achieved an effectiveness rate of 93.34%. These findings highlight the potential of ODRUN as an effective tool for solving complex optimization problems.

However, the research still has some limitations. For instance, research could extend the algorithm to tackle multi-stage decision-making problems, such as dynamic or sequential decision-making in uncertain environments in the future. Additionally, integrating the algorithm with machine learning techniques or deep learning models could provide further improvements. Furthermore, future work could apply this algorithm to other real-world problems, such as joint replenishment with stochastic demand, supply chain optimization, and the integration of optimization and scheduling in production environments.

References

[1] HOLLAND J H. Genetic algorithms. Scientific American, 1992, 267(1): 66–72.

[2] STORN R, PRICE K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341–359.

[3] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer. Proc. of the Advances in Engineering Software, 2014: 46−61.

[4] HAN M, DU Z, YUEN K F, et al. Walrus optimizer: a novel nature-inspired metaheuristic algorithm. Expert Systems with Applications, 2024, 239: 122413.

[5] AHMADIANFAR I, HEIDARI A A, GANDOMI A H, et al. RUN beyond the meta phor: an efficient optimization algorithm based on Runge Kutta method. Expert Systems with Applications, 2021, 181: 1115079.

[6] YILDIZ B S, MEHTA P, PANAGANT N, et al. A novel chaotic Runge Kutta optimization algorithm for solving constrained engineering problems. Journal of Computational Design and Engineering, 2022, 9(6): 2452–2465.

[7] YE J H, XIE L R, MA L, et al. A novel hybrid model based on Laguerre polynomial and multi-objective Runge–Kutta algorithm for wind power forecasting. International Journal of Electrical Power & Energy Systems, 2023, 146: 108726.
[8] HUANG J P, CHEN Y, HEIDARI A A, et al. Improved Runge Kutta optimization using compound mutation strategy in reinforcement learning decision making for feature selection. Journal of Bionic Engineering, 2024, 21(5): 2460–2496.
[9] ZHANG M L, CHEN H L, HEIDARI A A, et al. OCRUN: an oppositional Runge Kutta optimizer with cuckoo search for global optimization and feature selection. Applied Soft Computing, 2023, 146: 110664.
[10] QIAO Z L, LI L, ZHAO X C, et al. An enhanced Runge Kutta boosted machine learning framework for medical diagnosis. Computers in Biology and Medicine, 2023, 160: 106949.
[11] CHEN Z Q, KUANG F J, YU S D, et al. Static photovoltaic models' parameter extraction using reinforcement learning strategy adapted local gradient Nelder-Mead Runge Kutta method. Applied Intelligence, 2023, 53(20): 24106–24141.
[12] WANG Y J, ZHAO G H. A comparative study of fractional-order models for lithium-ion batteries using Runge Kutta optimizer and electrochemical impedance spectroscopy. Control Engineering Practice, 2023, 133: 105451.
[13] EL-SATTAR H A, KAMEL S, HASSAN M H, et al. Optimal sizing of an off-grid hybrid photovoltaic/biomass gasifier/battery system using a quantum model of Runge Kutta algorithm. Energy Conversion and Management, 2022, 258: 115539.
[14] NASSEF A M, HOUSSEIN E H, HELMY B E, et al. Optimal reconfiguration strategy based on modified Runge Kutta optimizer to mitigate partial shading condition in photovoltaic systems. Energy Reports, 2022, 8: 7242–7262.
[15] CHEN H L, AHMADIANFAR I, LIANG G X, et al. A successful candidate strategy with Runge-Kutta optimization for multi-hydropower reservoir optimization. Expert Systems with Applications, 2022, 209: 118383.
[16] WOLPERT D H, MACREADY W G. No free lunch theorems for optimization. IEEE Trans. on Evolutionary Computation, 1997, 1(1): 67–82.
[17] DEHGHANI M, MONTAZERI Z, TROJOVSKA E, et al. Coati optimization algorithm: a new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems, 2023, 259: 110011.
[18] AHMADIANFAR I, BOZORG-HADDAD O, CHU X. Gradient-based optimizer: a new metaheuristic optimization algorithm. Information Sciences, 2020, 540: 131–159.
[19] KUMAR A, PRICE K V, MOHAMED A W, et al. Problem definitions and evaluation criteria for the CEC 2022 special session and competition on single objective bound constrained numerical optimization. Singapore: Nanyang Technological University, 2022.
[20] QU H, AI X Y, WANG L. Optimizing an integrated inventory-routing system for multi-item joint replenishment and coordinated outbound delivery using differential evolution algorithm. Applied Soft Computing, 2020, 86: 105863.
[21] WANG S R, WANG L. Efficient methods for stochastic joint replenishment and delivery problem. International Transactions in Operational Research, 2022, 29(4): 2288–2315.
[22] KHOUJA M, GOYAL S. A review of the joint replenishment problem literature: 1989–2005. European Journal of Operational Research, 2008, 186(1): 1–16.
[23] CREEMERS S, BOUTE R. The joint replenishment problem: optimal policy and exact evaluation method. European Journal of Operational Research, 2022, 302(3): 1175–1188.
[24] LIU S Y, LIU O, JIANG X M. An efficient algorithm for the joint replenishment problem with quantity discounts, minimum order quantity and transport capacity constraints. Mathematics, 2023, 11(4): 1012.
[25] COELHO L C, LAPORTE G. Optimal joint replenishment, delivery and inventory management policies for perishable products. Computers & Operations Research, 2014, 47: 42–52.
[26] COHEN-HILLEL T, YEDIDSION L. The periodic joint replenishment problem is strongly NP-Hard. Mathematical Methods of Operations Research, 2018, 43(4): 1269–1289.
[27] CUI L, CHEN Y, TIAN Y, et al. Response strategies for coping with imperfect items of a joint replenishment model with the adaptive bare-bone differential evolution. Expert Systems with Applications, 2023, 225: 120091.
[28] CHEN Y R, YANG L Q, JIANG Y S, et al. Joint replenishment decision considering shortages, partial demand substitution, and defective items. Computers & Industrial Engineering, 2019, 127: 420–435.
[29] WANG L, WANG S R, GONG Y, et al. Optimizing a multi-echelon location-inventory problem with joint replenishment: a Lipschitz ϵ-optimal approach using Lagrangian relaxation. Computers & Operations Research, 2023, 151: 106128.
[30] ERSAHIN N, GIANNETTI M, HUANG R. Trade credit and the stability of supply chains. Journal of Financial Economics, 2024, 155: 103830.
[31] TARKOM A, YANG L. Presidential economic approval rating and trade credit. International Review of Financial Analysis, 2024, 93: 103236.
[32] CHANG C T, CHENG M C, SOONG P Y. Impacts of inspection errors and trade credits on the economic order quantity model for items with imperfect quality. International Journal of Systems Science: Operations & Logistics, 2016, 3(1): 34−48.