Swarm and Evolutionary Computation 86 (2024) 101486

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

A multi-swarm optimizer with a reinforcement learning mechanism for large-scale optimization

Xujie Wang [a], Feng Wang [a,*], Qi He [a], Yinan Guo [b]

[a] School of Computer Science, Wuhan University, Wuhan, 430072, China
[b] School of Mechanical Electronic and Information Engineering, China University of Mining and Technology (Beijing), Beijing 100083, China

ARTICLE INFO

Keywords:
Master–slave structure
Particle swarm optimization
Reinforcement learning
Large-scale global optimization

ABSTRACT

Large-scale global optimization (LSGO) problems involve numerous decision variables, are similar to real-world problems, and have generated research interest. To solve LSGO, a particle swarm optimizer (PSO) has been used. However, the many local optima and huge search space severely limit the effectiveness of the classic PSO. Dealing with the complexity of LSGO while avoiding the local optima is the main challenge of large-scale optimization algorithms. A multiswarm strategy has also been introduced to improve swarm diversity; however, it reduces the convergence speed. Previous studies have shown that reinforcement learning (RL) can improve the convergence ability of EAs owing to its increasing learning ability. In this study, we develop a multiswarm optimizer with an RL mechanism (MSORL) for LSGO. The MSORL includes a tri-particle group structure for subswarms to save the computational cost and balance the diversity and convergence. An RL-guided updating strategy is designed to enhance the convergence speed, and an adaptive tolerance-based search mechanism is employed to improve the diversity and avoid the local optima. The experimental results prove that the MSORL outperforms other state-of-the-art algorithms in terms of the convergence accuracy and speed.

1. Introduction

Large-scale global optimization (LSGO) has become a well-known research topic for evolutionary algorithms (EAs) and is gaining increasing research attention [1,2]. Different from traditional optimization problems, which typically involve 10–50 decision variables, LSGO problems are frequently influenced by thousands of decision variables. LSGO problems are widely found in real-life applications and have gained significant research interest [3–5], such as in oil well drilling distribution [6] and cloud workflow scheduling [7]. Generally, a minimization LSGO problem can be expressed as follows:

$$\begin{aligned} Min \quad & f(X), X = [x^1, x^2, \ldots, x^D] \\ & D \geq 500 \end{aligned} \tag{1}$$

where X belongs to a continuous search space with D dimensions.

The search space of an LSGO problem expands exponentially with increasing dimensionality; however, the required computational cost also rapidly increases. Traditional EAs do not easily rapidly converge to the global optimum, resulting in a low search efficiency. In addition, an LSGO problem is very complex and has numerous local optima. During optimization, traditional EAs are prone to repeatedly fall into the local optima, resulting in premature convergence. This phenomenon is known as the "curse of dimensionality". To overcome the challenges of a vast search space and numerous local optima, LSGO algorithms have been proposed. Numerous methods have been designed, which mainly can be categorized into dimensionality-reduction- and search-enhanced-based algorithms.

Most dimensionality-reduction-based algorithms divide LSGO problems into subproblems with fewer variables. Some of these algorithms perform decomposition before optimization, whereas others design dynamic reduction strategies. Among the algorithms that perform decomposition before optimization, some consider each variable as an independent subcomponent [8,9], whereas the others divide variables into groups based on their correlation [10,11]. In the algorithms that adopt dynamic reduction strategies, the subcomponents can be altered adaptively according to the problem [12–15]. By contrast, dimensionality-reduction strategies do not separate the problems [16]. However, the effectiveness of a dimensionality-reduction strategy is unstable for non-separable problems. In addition, dimensionality reduction incurs an additional computational cost, which lowers the search efficiency.

To address the above limitations, search-enhanced-based algorithms have been proposed. Some of these algorithms enhance the swarm diversity by reinitializing the swarm when it falls into the local optima [17,18], whereas the others adopt new updating strategies [19,20] or multiple strategies for different scenarios [21,22]. Moreover,

* Corresponding author.
E-mail address: fengwang@whu.edu.cn (F. Wang).

https://doi.org/10.1016/j.swevo.2024.101486
Received 16 May 2023; Received in revised form 9 December 2023; Accepted 19 January 2024
Available online 3 February 2024

special swarm structures have been designed for LSGO problems [23–25]. While these algorithms can solve non-separable LSGO problems more reliably, the massive search space still poses a challenge to the convergence speed and efficiency.

Reinforcement learning (RL) mechanisms have proven effective in improving the performance of EAs. RL-based evolutionary computation (RLEC) has shown significant potential [26]. RL can aid EAs in the particle update process [27,28], strategy selection, and parameter adjustment [29,30]. It also assists hyperheuristic algorithms in swarm structure adjustment [31,32]. With RL, EAs can learn from historical optimization information and self-adaptively adjust, which is promising for complex LSGO problems. However, the rapid convergence of RL can also cause premature problems.

To overcome the low search efficiency of traditional large-scale optimization algorithms and address the premature convergence issue in RL, this study combined a subswarm structure with RL and developed a multiswarm optimizer with an RL mechanism (MSORL). This approach effectively balances the convergence and diversity for LSGO problems. Different from other particle swarm optimizer (PSO) algorithms that integrate RL, the MSORL algorithm not only employs an RL-guided (RLG) updating strategy but also integrates a tri-particle group structure and an adaptive tolerance-based search (ATS) mechanism. The objectives are to enhance the population diversity and expedite the convergence process. Additionally, to avoid the premature problem, the RL mechanism introduced in MSORL uses the Softmax operator instead of ϵ-greedy as its policy. In addition, to fairly judge the difficulty of updating different particles, a specially designed reward function is also adopted.

(1) In the MSORL, master–slave swarm and tri-particle group structures are adopted for the subswarms. The tri-particle group structure for the subswarms selects the updating particles between generations and adopts an appropriate updating strategy that is suitable for exploitation or exploration. It also saves the computational cost.
(2) The RLG updating strategy self-adaptively guides the search directions of particles. Using the historical information obtained from all subswarms in the pregenerations, RLG can guide particles to explore the most promising area of the search space and improve its convergence speed.
(3) The ATS mechanism monitors the subswarm situations and helps avoiding the local optima. This mechanism reflects the convergence state of a subswarm and invokes a random search updating strategy to improve exploration when becoming trapped in the local optima. The ATS can effectively improve the diversity of the MSORL.

The remainder of this paper is organized as follows. Section 2 briefly introduces the related studies of LSGO algorithms and RLEC. Section 3 presents the MSORL in detail. The experimental results are presented and discussed in Section 4. Finally, we conclude our study and discuss future directions in Section 5.

2. Preliminary work

In this section, we comprehensively discuss the related studies of large-scale optimization and RLEC algorithms separately.

2.1. LSGO algorithms

To deal with LSGO problems efficiently, large-scale optimization algorithms must rapidly search through a vast and complex search space while maintaining the diversity to avoid the local optima. Extensive research has been conducted on large-scale evolutionary optimization algorithms, which mainly adopt two approaches: dimensionality reduction and search enhancement. In the following, a detailed introduction and analysis of these approaches are presented.

2.1.1. Dimensionality-reduction-based algorithms

There are two main dimensionality-reduction-based algorithms for LSGO problems with different reduction methods, i.e., divide-and-conquer approach and surrogate-assisted models.

Most dimensionality-reduction-based algorithms use a divide-and-conquer approach to divide LSGO problems into subproblems to reduce their complexity, and then solve the subproblems separately. Some variants determine the subproblem partitioning based on the correlation between different decision variables before the optimization process begins. The cooperative co-evolutionary algorithm proposed by Potter et al. adopts a cooperative co-evolutionary (CC) framework, treating all variables as individual subproblems and solving them separately [8]. In addition, for nonadditive decomposable problems, Chen et al. proposed a general separability grouping strategy [33]. However, with the increasing complexity of LSGO problems, static partitioning and dimensionality reduction strategies do not work effectively. Therefore, variants that allow dynamic adjustments of decomposition have been proposed [14]. A dynamic segment-based predominant learning swarm optimizer (DSPLSO) changes the grouping of its subproblems after every iteration that fails to update the global best solution [15].

Recently, some dimensionality-reduction-based algorithms utilize surrogate-assisted models, such as neural network, to expedite the fitness evaluation process for LSGO problems. Ullah et al. employed four dimensionality-reduction strategies: principal component analysis, kernel principal component analysis, autoencoders, and variational autoencoders. These techniques aim to reduce the dimensionality of the surrogate model [16]. Wang et al. proposed a progressive sampling surrogate-assisted PSO that utilizes a radial basis function network surrogate to efficiently solve LSGO problems [18]. Although this algorithm does not directly participate in dimensionality reduction, it employs a surrogate model to provide a computationally efficient representation of the objective.

During the reduction process, the solution space may partially change and inevitably lead to the loss of some original information [34]. For the algorithms using a divided-and-conquer approach, dividing LSGO problems into subproblems may cause the loss of crucial information because some correlated variables may be separated. For the algorithms using surrogate-assisted models, some redundant dimensions might be discarded, leading to information loss about the solution space. How to design a proper decomposition strategy is very important to the problem solving [10].

2.1.2. Search-enhanced-based algorithms

To avoid the above limitations of dimensionality-reduction-based algorithms, several search-enhanced large-scale optimization algorithms have been proposed. These algorithms adopt strategies such as reinitialization, optimized updating strategies, and special population structures to efficiently search the vast search space while maintaining the diversity to avoid the local optima.

A reinitialization strategy refers to generating new solutions by resampling to avoid the local optima. Cheng et al. improved the diversity of the algorithm by partially resetting the inactive areas of the population [17]. Wang et al. initiated a resetting strategy when the average distance between the particles in the population was below a preset threshold [18]. Although a reinitialization strategy effectively avoids falling into the local optima, it may affect the convergence process of the algorithm each time it is triggered and its timing is difficult to determine.

Designing more effective updating strategies to improve the search efficiency or diversity, and thus, improve the performance of algorithms, is also needed. Cheng et al. proposed a competitive swarm optimizer (CSO) [19] and a social learning PSO (SLPSO) [20], which allow particles to learn from a random better particle and the population mean position to maintain the diversity. In addition, Jian et al. introduced a region coding scheme for SLPSO [35]. Li et al. modified two learning exemplars of particles to address the exploration and

exploitation performance separately [36]. However, a fixed updating strategy does not easily meet the requirements of different situations in LSGO problems. Therefore, some algorithms use multiple updating strategies based on different situations [21,22,37]. However, for these algorithms, designing a unified mechanism to integrate all strategies is difficult.

Studies have also proposed large-scale optimization algorithms with various special population structures [23,24]. HCLPSO [2] algorithm utilizes two subpopulations and employs different strategies to maintain the swarm diversity and convergence performance well. Yang et al. divided a swarm into multiple levels and proposed a dynamical level-based learning swarm optimizer (DLLSO) [38]. Wang et al. introduced an RL mechanism in swarm structure adjustment [25]. The design of special population structures frequently needs to consider both the diversity and search efficiency of the population and must be constantly adaptively adjusted during the optimization process to achieve good performance.

A subswarm structure that improves the diversity of large-scale optimization algorithms has also been adopted in many studies. Some methods decompose LSGO problems and solve them separately. A cooperative PSO partitions a swarm into K subswarms before optimization [9]. However, a CC-PSO (CCPSO) [12] and CCPSO2 [13] involve adaptive strategies to dynamically control the division of subproblems. Some methods simply adopt a subswarm structure to improve the swarm diversity. Wang et al. introduced a master–slave population structure into large-scale PSO algorithms [39,40]. However, algorithms with subswarms increase the computational cost for each subswarm, which may decrease the search efficiency.

The above research on LSGO algorithms has significantly progressed in maintaining the population diversity and avoiding the local optima compared to traditional algorithms. However, to efficiently search the massive search space and deal with various LSGO problems, improving the search efficiency and generalization ability are also worth investigating.

2.2. RLEC

RL-based EAs use the historical data generated during optimization to train and adaptively adjust. This method has demonstrated good performance in solving various complex problems [26,41]. With RL, EAs can select the most reasonable strategy for the current state based on the expected digital reward. The following introduces the research related to RLEC.

Some RL-based EAs directly assist in problem optimization. Kalakrishnan et al. defined an objective function based on past optimization behavior [27]. RL with training on historical data can improve the convergence accuracy and effectively save the computational cost [28]. RL has proven to be a good complementary subcomponent in improving the convergence efficiency of EAs. However, it requires prior training and does not provide significant performance improvement before sufficient training.

Adapting an algorithm strategy dynamically according to the problem has always been effective in improving EAs. However, traditional strategies frequently lack flexibility; hence, EAs have introduced RL to achieve dynamic adaptive adjustment. Samma et al. proposed an RL-based memetic PSO, in which RL is applied to policy selection to balance the exploration and convergence behaviors [42]. RL is also applied to evaluate a new search area and parameter adjustment [29,30,43]. Applying RL to parameter and strategy adjustments can avoid the inefficiency of manual adjustment methods.

Research on RLEC extends beyond strategy adjustment, and it can also be applied to areas such as population structure adjustment and selection of hyperheuristic algorithms. Xu et al. designed multiple topological structures for populations and dynamically adjusted them using RL mechanisms [31]. Runarsson et al. utilized RL to evaluate and select subalgorithms in a hyperheuristic algorithm [32]. The learning ability of RL allowed the algorithm to appropriately adjust to various scenarios by learning from historical data. It also enhances the robustness of hyperheuristic and dynamic population structure algorithms, particularly in complex and long-term optimization processes.

Preliminary research indicates that RL mechanisms can assist EAs in dynamically adjusting their optimization direction, strategy selection, and even population structure. The use of RL can reduce the uncertainty of manual adjustment, improve the search efficiency of EAs, and balance the ability to explore and converge. The high convergence speed of RL may cause premature problems. Concurrently, a multiswarm PSO can maintain good diversity but shows poor convergence because each subswarm needs to be optimized separately. By combining RL and multiswarm PSO, the MSORL can overcome the massive search space and numerous local optima of LSGO problems.

3. MSORL algorithm

In the MSORL, a tri-particle group structure for subswarms, an RLG updating strategy, and an ATS mechanism are employed to help improve the search efficiency and avoid the local optima.

A brief flowchart of the MSORL is shown in Fig. 1. After the initialization, the subswarms are separated into sorted groups with three particles, i.e., tri-particle groups, before every generation. Each tri-particle group selects only one particle to be updated, x_{ij2} or x_{ij3}, are based on a probability that is obtained by the ATS mechanism. Different particles will adopt different updating strategies, RLG or ATS, which are introduced in Sections 3.2 and 3.3. The remainder of the tri-particle groups enter the next generation directly. After every generation, the master archive, F_g, Q-value, and T-value are updated accordingly. When the termination condition is satisfied, the best solution in the master archive is output.

3.1. Tri-particle group structure for subswarms

In the MSORL, a master–slave swarm structure is adopted to improve the diversity. The MSORL has one master node and NS slave subswarms with NP particles. Because particles in different slave subswarms barely exchange information and mainly interact with particles in the same group, the MSORL can search different areas simultaneously. The master node maintains an archive that contains NS particles, which are the particles with the best fitness values obtained from the subswarms. The particles in the master node do not update themselves but participate in the optimization by offering potential high-quality search directions for the RLG strategy.

Multi-swarm is helpful in locating multiple optimal solutions and can improve the search efficiency of LSGO. HCLPSO [2] is a typical multi-swarm optimizer which uses two subswarms to enhance exploration and exploitation simultaneously. Different from HCLPSO, MSORL employs multiple subswarms and uses a new tri-particle group structure in each subswarm to improve search efficiency.

In MSORL, the subswarms are divided into groups of three particles, and particles in each group are sorted by their fitness values. To retain the best historical experiences, the best particle is directly passed to the next generation. Concurrently, a search strategy in ATS is designed to determine whether the suboptimal and the worst particles can be updated, and more details will be introduced in Section 3.3. As better particles tend to be more efficient for exploitation, while worse particles often have greater potential for exploration, the tri-particle group structure employs an exploitation updating strategy and an exploration updating strategy to update the particles. If the suboptimal one is selected, this particle undergoes exploitation updating, whereas if the worst one is selected, it undergoes exploration updating. Only the selected particle is updated, and the remaining particles move directly to the next generation. In this case, each particle in the tri-particle groups has the opportunity to directly enter the next generation.

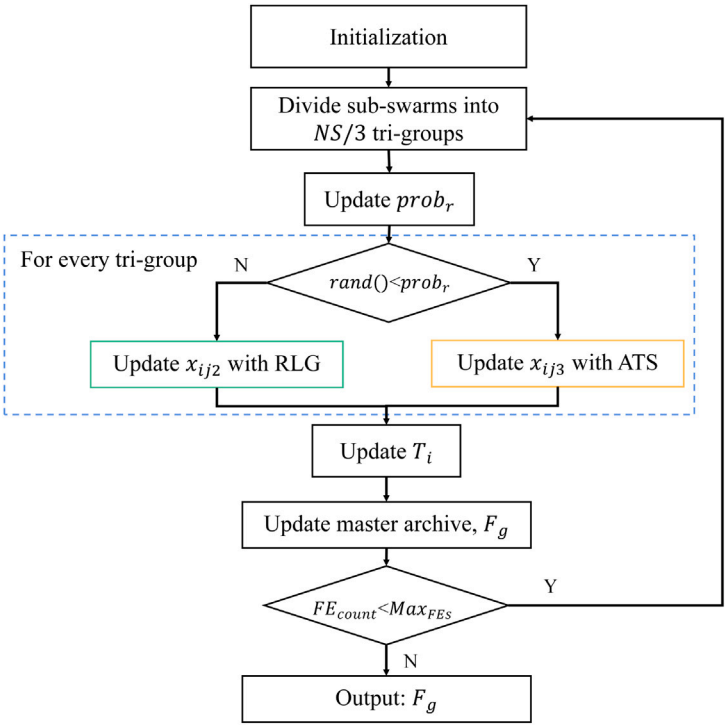


Fig. 1. The flowchart of MSORL.

It is well known that the pairwise competition mechanism using dual-particle structure has been commonly used in previous studies [19]. The algorithms using dual-particle structure update the worse particle and may slow down the convergence process to some extent. For LSGO, since the search space is more complex, in order to improve the search efficiency, we propose a new tri-particle group structure in MSORL. In this tri-particle group structure, there are 3 particles, which represent optimal solution, suboptimal solution, and the worst solution, respectively. And we group these three particles together and name this kind of structure in subswarms a 'tri-particle group structure'.

Since the tri-particle group structure allows the suboptimal particles to participate in the updating process, and only 1/3 of particles in subswarms are updated in each generation, it costs less time on fitness evaluation and can get faster convergence. But if the group contains more particles, the complexity of sorting the group will grow rapidly, which requires more specially designed updating strategies and may decrease the convergence performance.

As shown in Fig. 2, for the jth group in the ith subswarm, the particles are referred as x_{ij1}, x_{ij2}, and x_{ij3}. These three particles compose a 'tri-particle group', and they are sorted by fitness values. Let x_{ij1} be the best particle, x_{ij2} be the suboptimal one, and x_{ij3} be the worst one. From generation t to generation $t+1$, x_{ij1} is not updated to avoid the loss of fine solutions during the updating process. In addition, an ATS selection strategy decides whether to update x_{ij2} or x_{ij3} with a suitable updating strategy. The ATS chooses the updating particle based on the probability, $prob_b$; therefore, every particle has the opportunity to be preserved for the next generation. When a subswarm prefers exploitation behaviors, more poor particles are not updated to prevent the subswarm from being premature. Otherwise, more good particles are reserved for further convergence.

The main procedure of the tri-particle group structure for subswarms is described in Algorithm 1. It is a basic framework that demonstrates how each subswarm is updated in the MSORL. As shown in lines 4–9 in Algorithm 1, after being separated into tri-particle groups, each tri-particle group only updates x_{ij2} or x_{ij3}, whereas the remainder particles enter the next generation directly.

Algorithm 1 Tri-particle group structure for sub-swarms

Input: the i^{th} sub-swarm in the t^{th} generation $P_i(t)$
Output: the i^{th} sub-swarm in the $(t+1)^{th}$ generation $P_i(t+1)$
1: Divide the sub-swarms into groups of three particles randomly;
2: Sorted each group by fitness value;
3: **for** $j = 1 \rightarrow NP/3$ **do**
4: x_{ij1} enters the next generation directly;
5: Adopt a selection strategy for x_{ij2} and x_{ij3}:
6: **Case 1:** x_{ij2} is selected
7: Update x_{ij2} with a exploitation updating strategy and preserve x_{ij3} for the next generation;
8: **Case 2:** x_{ij3} is selected
9: Update x_{ij3} with a exploration updating strategy and preserve x_{ij2} for the next generation;
10: **end for**

Because the strategies embedded in the tri-particle group structure are replaceable, various updating strategies can be integrated into it. Different updating strategies for different updating particles and the selection strategy significantly influence the effectiveness of the structure. In the MSORL, the proposed RLG and ATS are selected to integrate into a tri-particle group structure. A selection strategy is also embedded in the ATS. More details are presented in the following subsections.

3.2. RLG updating strategy

The MSORL explores the search space more efficiently using the tri-particle group structure and maintains good diversity during optimization. However, the traditional updating strategies of PSO variants deteriorate rapidly in the vast search space of LSGOs. To enhance the convergence efficiency of the MSORL, an RLG updating strategy is proposed. With historical information accumulated between generations, the particles using RLG can more effectively detect the search space.

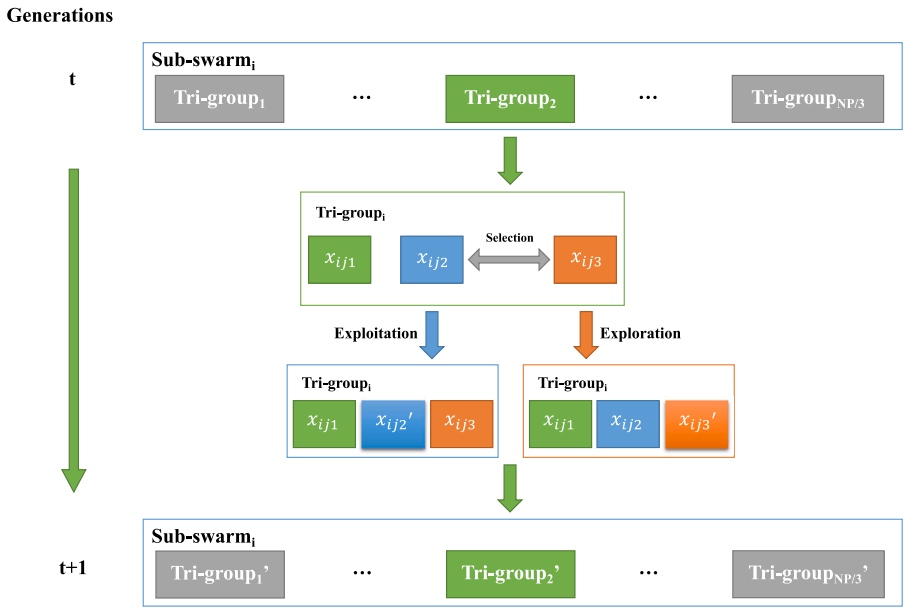


Fig. 2. The updating process of tri-particle group structure.

In the master–slave structure, particles in the same subswarm tend to concentrate in a similar search area. Therefore, they frequently have analogous reactions to different search directions. Additionally, the best particles of each subswarm represent the potential best learning exemplars of the area. In that case, in RLG, the best particles of the sub-swarms are stored in a master archive and considered as RL actions, whereas the subswarms are considered as states.

In the jth tri-particle group of the ith subswarm, when the second-best particle, x_{ij2}, is selected, it is updated with RLG. The updating particle select the best particles in its group x_{ij1} and one particle from the master archive as the learning exemplars. The updating process of the particles is expressed in Eq. (2) and (3):

$$v_{ij2}^d(t+1) = r_1 v_{ij2}^d(t) + r_2(x_{ij1}^d(t) - x_{ij2}^d(t)) + \phi_1 r_3(x_{Ar}^d(t) - x_{ij2}^d(t)) \tag{2}$$

$$x_{ij2}^d(t+1) = x_{ij2}^d(t) + v_{ij2}^d(t+1) \tag{3}$$

where $v_{ij2}^d(t)$ and $x_{ij2}^d(t)$ are the velocity and position of the second-best particle from the jth group of the ith subswarm on the dth dimension in the tth iteration. $x_{ij1}^d(t)$ is the best particle in the jth group. $x_{Ar}^d(t)$ is an exemplar chosen from the master archive. r_1, r_2, and r_3 represent three numbers that are randomly generated in the range $[0,1]$. ϕ_1, which restricts the influence of the exemplars from the master archive, is a parameter within [0,1].

In RLG, a Q-table is constructed to estimate the search potential of each particle in the master archive for the corresponding subswarms. For the particles that adopt RLG, a particle is selected from the master archive as a learning exemplar based on the Q-table for the updating process. The probabilities of the jth particle in the master archive to be chosen as an exemplar for the ith subswarm are determined with the following rules.

$$prob_{i,j} = \frac{e^{Q(i,j)}}{\sum_{q=0}^{NS} e^{Q(i,q)}} \tag{4}$$

where $Q(i,j)$ is the Q-value for selecting the jth particle in the master archive as the learning exemplar for the particles of the ith subswarms. NS is the size of the master archive. Instead of using the traditional $\epsilon-greedy$ method, which can lead to prematurity of the Q-table, assigning every particle in the archive a relatively reasonable probability of being chosen improves the diversity of the updating process.

A better solution is more difficult to improve, representing better effectiveness of the selected learning exemplar and vice versa. Therefore, after the kth particle in the ith subswarm is updated with RLG and chooses the jth particle in the master archive as the learning exemplar, a reward r_{ijk} is obtained. It is based on the optimization of the solution and the rank of the particle in the subswarm by the following rule:

$$r_{ijk} = \begin{cases} \frac{1}{rank_{i,k}} & f_{x_{ij2}}(t+1) < f_{x_{ij2}}(t) \\ -\frac{1}{rank_{i,k}} & f_{x_{ij2}}(t+1) \geq f_{x_{ij2}}(t) \end{cases} \tag{5}$$

where $rank_{i,k}$ is the fitness rank of the kth particle in the entire ith subswarm.

After the reward r_{ijk} is received, $Q(i,j)$ is updated instantly, as expressed in Eq. (6), to adjust the learning potential of the jth particle.

$$Q(i,j) = Q(i,j) + \alpha \cdot (r_{ijk} + \gamma \cdot Max(Q(i)) - Q(i,j)) \tag{6}$$

where $Max(Q(i))$ represents the ith the highest Q-value of the sub-swarms. α and γ are the learning rate and discount factor, respectively.

The initial master archive is constructed using the best particles from every initial subswarm. After every generation, if the best particle, $gbest_i$, of the ith is updated, the new $gbest_i$ is uploaded to the master archive. In the master archive, every uploaded particle is compared with a random particle. Only the better particle is preserved in the master archive. As the new particle is irrelevant compared with the former one, the Q-values of the old particle cannot apply to the new one. As shown in Fig. 3, when the master archive receives the new particle, to empty the useless historical information of the former particle, the Q-value of this particle is reinitialized and set at 0.

The RLG process can be summarized as Algorithm 2. As expressed in line 6 of Algorithm 2, the Q-table is frequently updated after every particle that adopts the RLG strategy is updated, which ensures effectiveness.

The historical information obtained from an entire subswarm aids the particles in adjusting their search directions based on the former experience and the search results of the neighboring particles. The experimental results in Section 4 prove that RLG significantly improves the convergence efficiency of the MSORL.

3.3. Adaptive tolerance-based search mechanism

To avoid trapping in the massive local optima of LSGOs by monitoring the optimization states of the subswarms, the MSORL adopts the ATS mechanism. For instance, when the $pbest$ of a subswarm is

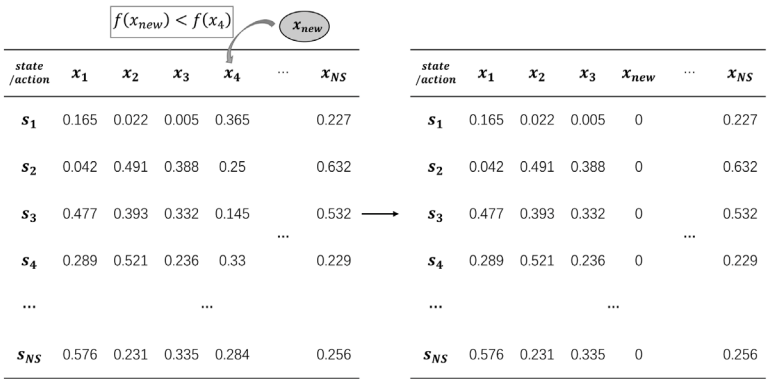


Fig. 3. The updating process of master archive and its Q-table.

Algorithm 2 Reinforcement learning guided updating strategy

Input: the swarm in the t^{th} generation $P(t)$
Output: the swarm in the $(t+1)^{th}$ generation $P(t+1)$
1: **for** $i = 1 \rightarrow NS$ **do**
2: **for** $j = 1 \rightarrow NP/3$ **do**
3: Use Eq. (4) to select a learning exemplar x_{Ar} from the master archive;
4: Update x_{ij2} with x_{Ar} and x_{ij1} as in Eq. (2) and Eq. (3);
5: Obtain reward r_{ijk} with Eq. (5);
6: Update the Q-table with r_{ijk} based on Eq. (6);
7: **end for**
8: **end for**
9: Update the master archive;

not updated for several generations, the subswarms are more probably to have encountered the local optima. A long stagnation implies a significant need of interference.

In the MSORL, an independent tolerance variable T, which is set at 0 initially, is used for every subswarm. After every generation, if *pbest* of the ith subswarm remains still, T_i will add one. Otherwise, the value of T_i is set at 0 again, such as Eq. (7).

$$T_i(t+1) = \begin{cases} 0 & f_{g_i}(t+1) < f_{g_i}(t) \\ T_i(t)+1 & f_{g_i}(t+1) = f_{g_i}(t) \end{cases} \tag{7}$$

where $T_i(t+1)$ is the T of the ith subswarm in the $t+1^{th}$ generation. In addition, g_i represents the ith best particle of the subswarm.

The ATS mechanism briefly represents the current optimization states of the related subswarms, i.e., it can monitor the state transformation of the subswarms and handle it accordingly. In a tri-particle group, the probability of updating the worst particle with the exploration strategy, $prob_r$, is controlled by the following rule (the value range of $prob_r$ is 0.1–1).

$$prob_r = 0.1 + 0.09 * Min(T_i, 10) \tag{8}$$

To ensure diversity during the search process, we assign the particles at least 10% probability of randomly exploring the search space. Furthermore, the probability increases linearly when the updating of g_i stagnates, reaching its peak when T_i reaches 10.

Additionally, when the worst particle of the jth tri-particle group in the ith subswarm, x_{ij3}, is chosen, it adopts a random search updating strategy and is updated by the following rules:

$$v^d_{ij3}(t+1) = r_1 v^d_{ij3}(t) + r_2(x^d_{r1}(t) - x^d_{ij3}(t)) + \phi_2 r_3(x^d_{r2}(t) - x^d_{ij3}(t)) \tag{9}$$

$$x^d_{ij3}(t+1) = x^d_{ij3}(t) + v^d_{ij3}(t+1) \tag{10}$$

where x_{r1} and x_{r1} are two random particles that are selected from all sub-swarms and $f(x_{r1}) < f(x_{r2}) < f(x_{ij3})$. ϕ_2 controls the influence of the worse learning exemplar.

Algorithm 3 Adaptive tolerance-based search mechanism

Input: the swarm in the t^{th} generation $P(t)$
Output: the swarm in the $(t+1)^{th}$ generation $P(t+1)$
1: **for** $i = 1 \rightarrow NS$ **do**
2: Update $prob_r$ with Eq. (8);
3: **for** $j = 1 \rightarrow NP/3$ **do**
4: **if** $rand() < prob_r$ **then**
5: Update x_{ij3} with x_{r1} and x_{r2} as in Eq. (9) and Eq. (10);
6: **else**
7: Update x_{ij2} with Algorithm 2;
8: **end if**
9: **end for**
10: Update T_i with Eq. (7);
11: **end for**

As shown in Algorithm 3, the ATS mechanism records the optimization situation of every subswarm. The T value influences the search preference of the selection procedure. From line 4 to line 8, when a subswarm falls into the local optima, the probability of selecting x_{ij3}, which adopts the random search updating strategy to be updated, will increase rapidly. As the random search updating strategy allows particles to learn from all the subswarms, it can ensure the diversity of the MSORL and avoid the local optima.

3.4. Procedure of MSORL

The main framework of the MSORL is elaborated in Algorithm 4. The Q-table is updated after every particle is updated. The T-value is updated after a subswarm is updated. The master archive is only updated between generations.

In Algorithm 4, the MSORL adopts a tri-particle group structure to determine the suitable updating strategy for different situations, which balances the search efficiency and diversity. Concurrently, from line 12 to line 14, Algorithm 2 is introduced to improve the convergence efficiency. It guides the particles to learn from the most promising exemplars. Additionally, from line 10 to line 11, the random search behavior in Algorithm 3 also helps enhance the diversity and avoids becoming trapped in the local optima.

In the MSORL, because only the three particles in the tri-particle groups need to be sorted, the time complexity is $O(NS \times NP)$. The time complexity of the updating process is $O(NS \times NP \times D)$, and that of the RL mechanism is $O(NS \times NP)$. Therefore, the overall time complexity of the MSORL is $O(NS \times NP \times D)$. In addition, the spatial complexity of the swarm is $O(NS \times NP \times D)$, and the spatial complexity of the Q

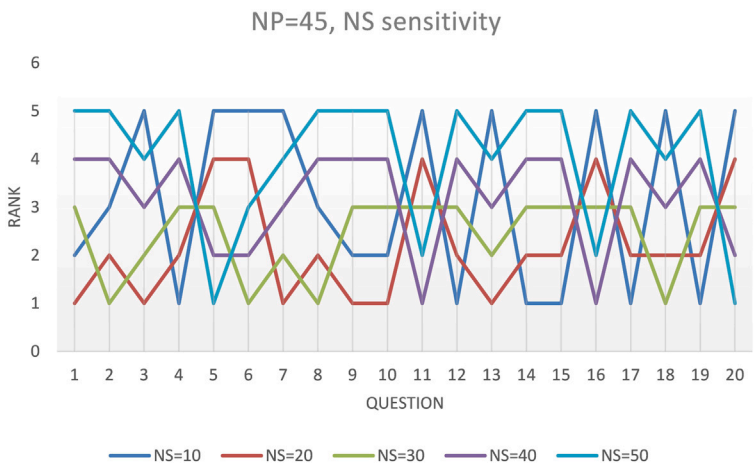


Fig. 4. NS sensitivity experiments.

Algorithm 4 Framework of MSORL

Input: swarm size NP, sub-swarm number NS
Output: the swarm best solution F_g

Initialize positions, velocities, and fitness value of all particles;
Initialize master archive as in Algorithm 2;
Initialize $FE_{count} = 0$, $T_{i \to NS} = 0$;
Initialize F_g is the best particle in the master archive;
while stop condition is not reached **do**
  for $i = 1 \to NS$ **do**
    Divide the i^{th} sub-swarm into $NS/3$ tri-particle groups by Algorithm 1;
    Use Eq. (8) to update $prob_r$;
    for $j = 1 \to NS/3$ **do**
      if $rand() < prob_r$ **then**
        Update x_{ij3} with Algorithm 3;
      else
        Update x_{ij2} and the Q-table with Algorithm 2;
      end if
    end for
    Update T_i by Eq. (7);
  end for
  Update the master archive based on Algorithm 2;
  Update F_g;
end while

table is $O(NS \times NP)$. Therefore, the overall spatial complexity of the MSORL is $O(NS \times NP \times D)$.

4. Experimental studies

We validated the effectiveness of the MSORL on two well-known test suites of LSGO: IEEE CEC2010 [44] and IEEE CEC2013 LSGO test suites [45]. In addition, ablation experiments have proven the validity of all adopted strategies.

4.1. Experimental setup

As two of the most well-known LSGO test suites, CEC 2010 and CEC2013 contain 6 fully separable functions ($f1$–$f3$, $f21$–$f23$), 23 partially separable functions ($f4$–$f18$, $f24$–$f31$), 3 overlapping functions ($f32$ to $f34$), and 3 non-separable functions ($f19$ to $f20$, $f35$). All of them are with 1000 variables. The experimental results on those comprehensive and well-designed functions can validate the effectiveness of our proposed algorithm.

In the MSORL, following the previous research, ϕ_1 and ϕ_2 were both set as 0.4 [19,38]. α and γ in Eq. (6) were set at 0.4 and 0.8, respectively [25,46].

Additionally, the time of fitness evaluation, Max_{FEs}, was set at 3,000,000. To ensure fair and comprehensive comparisons, the mean results and standard deviation (Std) values were obtained from 30 independent runs. Moreover, for the statistical significance, a Wilcoxon rank-sum test was performed, and the significance level, α, was set at 0.05. All algorithms run on a PC that has Intel(R) Core(TM) i7-4790 3.60-GHz CPU, 8-GB memory, and Windows 8.1 64-bit system.

4.2. Parameter sensitivity experiment and analysis

To test the appropriate parameter setting of the MSORL, parameter sensitivity experiments of the subswarm size, NP, and the subswarm number NS were conducted. To control the appropriate size of the swarm, NP belongs to $\{10, 30, 45, 60, 75\}$ and NS is in $\{10, 20, 30, 40, 40\}$. The test suite was CEC 2010 LSGO. The results of the different values are shown in Figs. 4 and 5. The x-axis has different functions and the y-axis represents the ranking among all values.

From Figs. 4 and 5, the average rankings of all the NP values are ranked as follows: 3.9, 2.7, 2.15, 2.8, and 3.45. The average rankings of the NS values are 3.15, 2.2, 2.45, 3.2, and 4. In addition, when NP is set at 45 or NS is set at 20, the MSORL both reaches the best result on 6 functions, which is the most compared with other parameter settings. Such settings have shown outstanding performance on both average and best results. Therefore, in the following experiments, NP was set at 45, and NS was set at 20 in the MSORL.

4.3. Comparison of results and analysis

The following eight state-of-the-art algorithms were selected for comparison, namely, DECC-DG [10], DLLSO [38], CCPSO2 [13], SL-PSO [20], CSO [19], SDLSO [7], RLLPSO [25], and LDE [47]. Among them, CCPSO2 and DECC-DG are dimensionality-reduction-based algorithms. DLLSO, SL-PSO, CSO, and SDLSO are search-enhanced-based algorithms. RLLPSO and LDE are RLEC algorithms. Comparing the different types of large-scale optimization and RLEC algorithms showed the effectiveness of the MSORL. For reasonableness, we utilized the compared algorithms with their recommended parameter. Table 1 provides the details of the parameter settings.

The results on the convergence accuracy of the MSORL and the comparison algorithms of CEC2010 and CEC2013 are presented in Tables 2 and 3, respectively. The best results are labeled with a bold font, whereas the suboptimal results are labeled with underlines. In addition, the overall $+/-/\approx$ results are also presented.

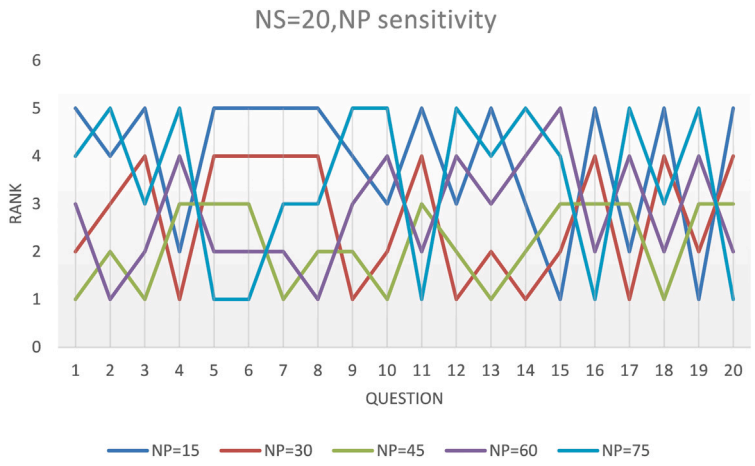


Fig. 5. NP sensitivity experiments.

Table 1
Comparison algorithms' parameters settings.

Algorithms	Parameters settings
DLLSO	$NP = 500,\ \phi = 0.4$
CCPSO2	$NP = 30,\ p = 0.5$
SL-PSO	$NP = 200,\ \epsilon = 0.1,\ \alpha = 0.5,\ \beta = 0.01$
CSO	$NP = 500,\ \phi = 0.1$
DECC-DG	$NP = 50,\ \epsilon = 0.001$
SDLSO	$NP = 400$
RLLPSO	$NP = 500,\ \phi = 0.4$
LDE	$NP = 50$

Tables 2 and 3 show that the MSORL outperforms all the compared algorithms on 15 benchmark functions in the convergence accuracy. In contrast, the remaining compared algorithms perform best on a maximum of 6 functions. Overall, the MSORL outperforms the other algorithms in terms of the convergence accuracy.

Compared to the dimensionality-reduction algorithms, DECC-DG and CCPSO2, MSORL has better results on 25 and 26 functions in 35 benchmark functions, respectively. Among the separable functions, CCPSO2 shows the best results on three functions, whereas MSORL shows better results than the other two algorithms on another three separable functions. The efficient convergence performance of the RLG strategy makes the MSORL perform outstandingly in terms of the convergence accuracy. The performance of the MSORL on separable functions is comparable to that of the dimensionality-reduction algorithms. Furthermore, as the non-separability of the benchmark functions gradually increases, CCPSO2 and DECC-DG start to deteriorate. On the contrary, as the functions become more complex, the superiority of the MSORL becomes more evident.

Compared to the search-enhanced-based large-scale optimization algorithms, such as DLLSO, SL-PSO, CSO, and SDLSO, the MSORL shows ascendant performances. In 35 test functions, the MSORL performs better on 21, 28, 24, and 21 functions, respectively. As for RLECs, MSORL has better performances on 19 and 27 functions compared with the RLLPSO and LDE. The master–slave swarm structure ensures the diversity of the MSORL, making it outperform LDE in most cases. Clarify that LDE, which is designed for regular optimization problems, falls into the local optima easily and converges at a very early stage. As the RLG strategy participates in the optimization process directly and improves the convergence efficiency, MSORL also shows better convergence accuracy than RLLPSO.

In terms of the convergence accuracy, MSORL outperforms all the compared algorithms. The RLG strategy improves the convergence efficiency of the algorithm, whereas the ATS mechanism, multi-swarm strategy, and tri-particle group structure in the subswarms ensure good diversity of the algorithm when dealing with complex problems. Consequently, the efficiency of the MSORL on separable functions was comparable to that of the algorithms that adopt dimensionality-reduction strategies. When facing more complex problems, MSORL outperforms the other algorithms in most functions. In summary, based on the experimental results, we conclude that the MSORL has better convergence accuracy than the other compared algorithms.

In addition to the requirement for convergence accuracy, the algorithms for LSGO problems must consider the convergence speed. To testify to the convergence speed of the MSORL, we plotted its convergence process and the comparison algorithm on all test functions. Among them, Fig. 6 shows the optimization processes on all the separable functions. Figs. 7 and 8 are for partially separable functions. In Figs. 9 and 10, the optimization of the most complex overlapping function and completely inseparable function are displayed. It should be noted here, the convergence speed results of LDE are not shown in the figures. As the LDE requires extra time to train the artificial neural network before optimization, it takes a much longer time to finish the search process than the other algorithms.

Based on the above figures, owing to the training cost of RL, the convergence speed of the MSORL is ordinary at the beginning of optimization. The master–slave structure, which ensures the diversity of the MSORL, requires subswarms to search the search space separately, influencing the convergence speed. Therefore, the MSORL does not outperform other algorithms in some of the functions. However, as the complexity of the functions increases, the convergence speed of the other comparative algorithms gradually declines. By contrast, the MSORL maintains its convergence speed owing to the effectiveness of the RLG updating strategy. Figs. 7 and 8 show that the high complexity of a function implies a lower convergence speed of the comparative algorithms. However, the convergence speed of the MSORL is stable and high, and its overall convergence performance remains among the top-three of the comparative algorithms. Moreover, the MSORL demonstrates a good and sustained convergence ability, frequently maintaining fast convergence speed even when the comparative algorithms are gradually converging or slowing down their convergence speed, as the results shown on $f8$, $f12$, and $f14$. Overall, the MSORL shows a clear advantage in the convergence speed in more complex LSGO problems, and its convergence speed also demonstrates good competitiveness in the separable functions.

Furthermore, the results on functions such as $f2$, $f8$, $f10$, and $f22$ demonstrate that the ATS mechanism ensures good diversity of the MSORL, enabling it to escape the local optima compared to the other algorithms. After the swarm has fallen into the local optima, the probability of performing random searches gradually increases, and the

Table 2
Comparison results on CEC2010 LSGO.

Functions	MSORL Mean±Std	DECC-DG Mean±Std	DLLSO Mean±Std	CCPSO2 Mean±Std	SL-PSO Mean±Std
$f1$	7.90E−22 ± 1.10E−22	5.04E+04 ± 2.26E+05(+)	2.68E−22 ± 5.30E−23(−)	3.25E+00 ± 4.42E+00(+)	8.26E−18 ± 1.51E−18(+)
$f2$	6.66E+02 ± 2.74E+01	4.38E+03 ± 1.55E+02(+)	9.45E+02 ± 3.80E+01(+)	**1.41E+02 ± 1.27E+02(−)**	1.90E+03 ± 1.37E+02(+)
$f3$	3.47E−14 ± 2.70E−15	1.68E+01 ± 2.02E−01(+)	6.87E−14 ± 4.14E−15(+)	3.37E−04 ± 6.59E−04(+)	1.77E+00 ± 2.86E−01(+)
$f4$	**1.38E+11 ± 2.33E+10**	3.60E+12 ± 9.78E+11(+)	8.09E+11 ± 2.05E+11(+)	4.28E+12 ± 2.78E+12(+)	5.11E+11 ± 1.76E+11(+)
$f5$	5.57E+06 ± 1.68E+06	9.07E+07 ± 1.83E+07(+)	1.29E+07 ± 2.47E+06(+)	3.90E+08 ± 6.41E+07(+)	2.94E+07 ± 7.03E+06(+)
$f6$	1.15E+00 ± 4.69E−01	3.85E+04 ± 2.07E+05(+)	2.95E−01 ± 6.38E−01(−)	1.74E+07 ± 4.56E+06(+)	1.98E+01 ± 4.03E+00(+)
$f7$	**5.89E−04 ± 3.47E−04**	2.53E+04 ± 1.30E+04(+)	6.33E+00 ± 8.97E+00(+)	1.32E+10 ± 1.30E+10(+)	6.23E+04 ± 3.96E+04(+)
$f8$	**2.38E+04 ± 1.05E+04**	4.63E+07 ± 2.64E+07(+)	2.33E+07 ± 1.67E+05(+)	7.25E+07 ± 7.25E+07(+)	1.03E+07 ± 6.45E+06(+)
$f9$	**1.28E+07 ± 1.06E+06**	5.43E+07 ± 3.52E+06(+)	4.03E+07 ± 3.96E+06(+)	1.12E+08 ± 2.16E+07(+)	2.87E+07 ± 2.90E+06(+)
$f10$	**5.73E+02 ± 3.05E+01**	4.23E+03 ± 1.33E+02(+)	8.82E+02 ± 4.27E+01(+)	4.57E+03 ± 3.32E+02(+)	2.51E+03 ± 1.58E+02(+)
$f11$	1.86E+01 ± 2.19E+00	1.29E+01 ± 5.78E−01(−)	4.96E+00 ± 4.87E+00(−)	1.99E+02 ± 4.15E+00(+)	2.26E+01 ± 2.00E+00(+)
$f12$	**5.56E+03 ± 4.93E+02**	1.58E+04 ± 2.23E+03(+)	1.45E+04 ± 1.00E+03(+)	1.70E+05 ± 1.80E+04(+)	1.99E+04 ± 1.21E+04(+)
$f13$	**5.51E+02 ± 1.36E+02**	8.60E+03 ± 3.87E+03(+)	9.65E+02 ± 3.11E+02(+)	1.49E+03 ± 6.04E+02(+)	1.12E+03 ± 6.29E+02(+)
$f14$	**4.01E+07 ± 2.90E+06**	3.07E+08 ± 1.39E+07(+)	1.33E+08 ± 6.86E+06(+)	4.09E+08 ± 1.12E+08(+)	8.34E+07 ± 6.80E+06(+)
$f15$	9.80E+03 ± 5.52E+01	6.07E+03 ± 1.50E+02(−)	1.44E+03 ± 2.27E+03(−)	9.05E+03 ± 4.23E+02(−)	1.12E+04 ± 1.06E+02(+)
$f16$	4.49E+00 ± 3.73E+00	1.49E−01 ± 3.82E−01(−)	3.20E+00 ± 1.83E+00(≈)	3.94E+02 ± 2.18E+00(+)	2.14E+01 ± 7.01E+00(+)
$f17$	1.95E+05 ± 1.71E+04	1.26E+05 ± 4.44E+03(−)	8.15E+04 ± 4.34E+03(−)	4.04E+05 ± 1.11E+05(+)	8.30E+04 ± 1.33E+04(−)
$f18$	**1.37E+03 ± 3.31E+02**	3.21E+09 ± 8.81E+08(+)	2.58E+03 ± 7.34E+02(+)	1.21E+04 ± 7.34E+03(+)	3.02E+03 ± 8.75E+02(+)
$f19$	6.62E+06 ± 3.39E+05	2.17E+06 ± 1.22E+05(−)	1.79E+06 ± 1.07E+05(−)	2.13E+06 ± 3.52E+06(−)	5.03E+06 ± 7.37E+05(−)
$f20$	1.23E+03 ± 7.36E+01	9.18E+10 ± 8.99E+09(+)	1.70E+03 ± 1.59E+02(+)	1.54E+03 ± 1.88E+02(+)	1.75E+03 ± 1.07E+02(+)
+/−/≈	–	15/5/0	13/6/1	17/3/0	18/2/0
Functions	MSORL Mean±Std	CSO Mean±Std	SDLSO Mean±Std	RLLPSO Mean±Std	LDE Mean±Std
$f1$	7.90E−22 ± 1.10E−22	4.70E−12 ± 6.67E−13(+)	**3.07E−23 ± 9.40E−24(−)**	4.48E−22 ± 1.70E−22(−)	8.95E+05 ± 1.78E+06(+)
$f2$	6.66E+02 ± 2.74E+01	7.55E+03 ± 2.56E+02(+)	7.13E+02 ± 3.86E+01(+)	9.72E+02 ± 5.71E+01(+)	7.59E+04 ± 4.23E+03(+)
$f3$	3.47E−14 ± 2.70E−15	2.60E−09 ± 1.68E−10(+)	**2.49E−14 ± 1.30E−15(−)**	8.67E−14 ± 1.08E−14(+)	2.10E+01 ± 1.22E−02(+)
$f4$	**1.38E+11 ± 2.33E+10**	1.31E+12 ± 1.89E+11(+)	4.66E+11 ± 1.06E+11(+)	6.50E+11 ± 5.02E+10(+)	1.15E+12 ± 3.05E+11(+)
$f5$	5.57E+06 ± 1.68E+06	4.22E + 06 ± 1.66E + 06(≈)	**4.05E+06 ± 1.99E+06(−)**	1.26E+07 ± 2.64E+06(+)	1.27E+08 ± 2.56E+07(+)
$f6$	1.15E+00 ± 4.69E−01	**8.03E−07 ± 2.27E−08(−)**	2.16E+01 ± 5.99E−03(+)	3.35E−01 ± 5.17E−01(−)	2.08E+07 ± 8.69E+04(+)
$f7$	**5.89E−04 ± 3.47E−04**	1.15E+04 ± 3.64E+03(+)	2.38E−01 ± 1.92E−01(+)	2.37E−01 ± 1.35E−01(+)	3.11E+05 ± 3.84E+05(+)
$f8$	**2.38E+04 ± 1.05E+04**	3.86E+07 ± 6.29E+04(+)	2.19E+05 ± 3.82E+04(+)	1.43E+07 ± 1.13E+07(+)	8.41E+07 ± 4.14E+07(+)
$f9$	**1.28E+07 ± 1.06E+06**	6.25E+07 ± 5.28E+06(+)	2.51E+07 ± 2.11E+06(+)	3.00E+07 ± 1.13E+06(+)	1.65E+08 ± 5.10E+07(+)
$f10$	**5.73E+02 ± 3.05E+01**	9.60E+03 ± 8.34E+01(+)	6.27E+02 ± 3.78E+01(+)	9.00E+02 ± 5.79E+01(+)	7.35E+04 ± 4.61E+03(+)
$f11$	1.86E+01 ± 2.19E+00	**4.23E−08 ± 5.81E−09(−)**	3.03E+01 ± 1.32E+01(+)	2.41E+00 ± 3.11E+00(−)	2.36E+02 ± 1.29E−01(+)
$f12$	**5.56E+03 ± 4.93E+02**	5.39E+05 ± 5.98E+04(+)	9.09E+03 ± 1.09E+03(+)	1.12E+04 ± 6.09E+02(+)	1.07E+05 ± 8.77E+03(+)
$f13$	**5.51E+02 ± 1.36E+02**	1.08E+03 ± 4.89E+02(+)	8.47E+02 ± 5.64E+02(+)	9.62E+02 ± 1.44E+02(+)	4.62E+04 ± 2.99E+04(+)
$f14$	**4.01E+07 ± 2.90E+06**	2.87E+08 ± 1.55E+07(+)	7.34E+07 ± 4.82E+06(+)	1.11E+08 ± 2.32E+06(+)	4.83E+08 ± 3.34E+07(+)
$f15$	9.80E+03 ± 5.52E+01	1.00E+04 ± 6.72E+01(+)	9.71E+03 ± 6.44E+01(−)	**8.41E+02 ± 4.71E+01(−)**	6.84E+04 ± 6.28E+03(+)
$f16$	4.49E+00 ± 3.73E+00	**5.88E−08 ± 6.89E−09(−)**	3.64E+01 ± 1.81E+01(+)	1.88E+00 ± 3.02E+00(−)	4.29E+02 ± 1.38E+00(+)
$f17$	1.95E+05 ± 1.71E+04	2.26E+06 ± 1.65E+05(+)	1.14E+05 ± 7.44E+03(−)	**5.31E+04 ± 1.63E+03(−)**	2.63E+05 ± 1.33E+04(≈)
$f18$	**1.37E+03 ± 3.31E+02**	3.05E+03 ± 2.79E+03(+)	2.11E+03 ± 8.40E+02(+)	2.10E+03 ± 3.90E+02(+)	1.06E+08 ± 1.59E+08(+)
$f19$	6.62E+06 ± 3.39E+05	9.85E+06 ± 5.18E+05(+)	3.27E+07 ± 2.87E+06(+)	**8.71E+05 ± 2.31E+04(−)**	1.40E+06 ± 6.92E+04(−)
$f20$	1.23E+03 ± 7.36E+01	**1.01E+03 ± 4.14E+01(−)**	1.17E+03 ± 9.75E+01(−)	1.84E+03 ± 1.68E+02(+)	3.41E+08 ± 9.36E+08(+)
+/−/≈	–	15/4/1	14/6/0	13/7/0	18/1/1

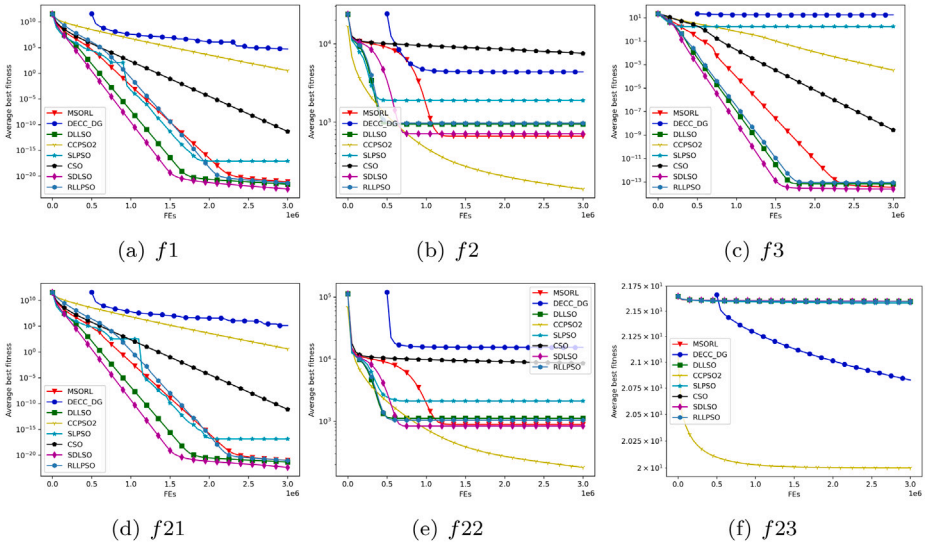
population can explore a wider solution space, ultimately allowing it to avoid the local optima. Additionally, for functions such as $f4$, $f7$, $f14$, $f24$, and $f27$, in the later stages of optimization, the convergence efficiency of most algorithms is affected by the randomness in their updating strategies, resulting in a gradual decrease in the convergence efficiency. However, the MSORL, which adopts the RLG strategy and learns from historical optimization experience, can maintain or even improve its convergence speed. In the optimization process of the MSORL, the tri-particle groups in the subswarm balance exploitation and exploration well, making its overall convergence curve smoother than those of other algorithms. This reflects the more stable nature of the MSORL. In conclusion, the convergence speed of the MSORL always maintains good competitiveness compared to the other algorithms, and it is more stable and less affected by the problem complexity.

As shown in Fig. 6, the MSORL algorithm converges more slowly than others in problems with separable decision variables. A possible reason is that MSORL requires extra computational time for the reinforcement learning process. As a result, for those LSGO problems with separable decision variables, it converges slower than others during the whole evolutionary process. Meanwhile, for those LSGO problems with non-separable decision variables, since the solution space is much more complex, in the early stage of the evolutionary process, MSORL may also be slightly slower than some other algorithms, but due to the RLG strategy, MSORL gradually improves the convergence speed and achieves better convergence performance ultimately.

The experimental results show that the MSORL is efficient in separable, partially separable, and non-separable LSGO problems, demonstrating good generalization ability. Furthermore, as the problems become

Table 3
Comparison results on CEC2013 LSGO.

Functions	MSORL Mean±Std	DECC-DG Mean±Std	DLLSO Mean±Std	CCPSO2 Mean±Std	SL-PSO Mean±Std
$f21$	1.03E−21 ± 2.21E−22	1.29E+05 ± 5.44E+05(+)	4.54E−22 ± 2.13E−22(−)	3.99E+00 ± 5.62E+00(+)	1.37E−17 ± 7.28E−18(+)
$f22$	8.85E+02 ± 5.35E+01	1.54E+04 ± 5.35E+01(+)	1.11E+03 ± 7.49E+01(+)	**1.80E+02 ± 1.27E+02(−)**	2.11E+03 ± 1.55E+02(+)
$f23$	2.16E+01 ± 5.71E−03	2.08E+01 ± 1.02E−02(−)	2.16E+01 ± 4.19E−03(≈)	**2.00E+01 ± 5.42E−04(−)**	2.16E+01 ± 2.80E−02(−)
$f24$	**1.31E+09 ± 3.67E+08**	1.79E+11 ± 5.88E+10(+)	6.30E+09 ± 1.40E+09(+)	3.10E+10 ± 1.27E+10(+)	4.27E+09 ± 1.25E+09(+)
$f25$	**5.19E+05 ± 7.51E+04**	5.34E+06 ± 4.43E+05(+)	6.69E+05 ± 1.09E+05(+)	1.48E+07 ± 4.55E+06(+)	8.26E+05 ± 1.13E+05(+)
$f26$	1.06E+06 ± 9.23E+02	1.06E+06 ± 1.41E+03(−)	1.06E+06 ± 6.45E+02(≈)	**1.04E+06 ± 6.87E+03(−)**	1.06E+06 ± 9.89E+02(≈)
$f27$	**3.36E+05 ± 1.74E+05**	1.35E+09 ± 3.66E+08(+)	1.61E+06 ± 8.14E+05(+)	8.28E+08 ± 1.25E+09(+)	1.57E+06 ± 6.79E+05(+)
$f28$	**3.72E+13 ± 1.23E+13**	7.79E+15 ± 2.08E+15(+)	1.19E+14 ± 4.18E+13(+)	2.34E+15 ± 2.00E+15(+)	9.42E+13 ± 4.23E+13(+)
$f29$	8.67E+07 ± 2.25E+07	1.27E+09 ± 1.75E+08(+)	1.31E+08 ± 3.84E+07(+)	3.74E+09 ± 1.02E+09(+)	8.72E+07 ± 2.82E+07(≈)
$f30$	9.40E+07 ± 2.85E+05	9.38E+07 ± 3.45E+05(−)	9.41E+07 ± 1.76E+05(≈)	**9.27E+07 ± 5.91E+05(−)**	9.32E + 07 ± 1.42E + 06(≈)
$f31$	9.57E+11 ± 3.81E+10	9.21E+11 ± 3.55E+09(−)	9.30E+11 ± 9.10E+09(−)	9.40E+11 ± 2.83E+10(≈)	9.34E+11 ± 1.70E+10(≈)
$f32$	1.32E+03 ± 8.91E+01	8.76E+10 ± 1.04E+10(+)	1.80E+03 ± 1.94E+02(+)	1.49E+03 ± 1.58E+02(+)	1.79E+03 ± 1.43E+02(+)
$f33$	**2.13E+08 ± 1.10E+08**	2.38E+10 ± 5.72E+09(+)	3.21E+08 ± 1.51E+08(+)	7.03E+09 ± 2.71E+09(+)	4.07E+08 ± 2.41E+08(+)
$f34$	**4.69E+07 ± 1.07E+07**	1.16E+11 ± 3.49E+10(+)	1.12E+08 ± 9.18E+07(+)	2.31E+11 ± 1.34E+11(+)	6.21E+08 ± 7.52E+08(+)
$f35$	2.41E+07 ± 1.90E+06	1.37E+07 ± 2.11E+06(−)	4.37E+06 ± 2.98E+05(−)	1.79E+07 ± 2.07E+07(−)	5.89E+07 ± 7.68E+06(+)
+/−/≈	–	10/5/0	8/4/3	9/5/1	10/1/4
Functions	MSORL Mean±Std	CSO Mean±Std	SDLSO Mean±Std	RLLPSO Mean±Std	LDE Mean±Std
$f21$	1.03E−21 ± 2.21E−22	7.32E−12 ± 1.17E−12(+)	**4.28E−23 ± 1.16E−23(−)**	9.01E−22 ± 3.50E−22(−)	2.07E+06 ± 6.80E+06(+)
$f22$	8.85E+02 ± 5.35E+01	8.57E+03 ± 1.60E+02(+)	8.33E+02 ± 5.03E+01(−)	1.04E+03 ± 7.40E+01(+)	2.28E+04 ± 1.01E+03(+)
$f23$	2.16E+01 ± 5.71E−03	2.16E+01 ± 6.57E−03(≈)	2.16E+01 ± 7.09E−03(≈)	2.16E+01 ± 6.21E−03(≈)	2.11E+01 ± 6.63E−02(−)
$f24$	**1.31E+09 ± 3.67E+08**	1.30E+10 ± 2.60E+09(+)	4.10E+09 ± 6.72E+08(+)	4.44E+09 ± 4.99E+08(+)	1.20E+10 ± 3.13E+09(+)
$f25$	**5.19E+05 ± 7.51E+04**	5.88E+05 ± 1.02E+05(+)	6.23E+05 ± 8.87E+04(+)	7.25E+05 ± 1.01E+05(+)	6.84E+06 ± 9.10E+05(+)
$f26$	1.06E+06 ± 9.23E+02	1.06E+06 ± 1.46E+03(≈)	1.06E+06 ± 9.04E+02(≈)	1.06E+06 ± 9.95E+02(≈)	1.06E+06 ± 2.15E+03(−)
$f27$	**3.36E+05 ± 1.74E+05**	5.10E+06 ± 1.77E+06(+)	1.21E+06 ± 8.58E+05(+)	9.84E+05 ± 2.47E+05(+)	4.19E+07 ± 1.34E+07(≈)
$f28$	**3.72E+13 ± 1.23E+13**	2.44E+14 ± 6.11E+13(+)	1.05E+14 ± 2.69E+13(+)	8.56E+13 ± 1.07E+13(+)	1.16E+14 ± 3.58E+13(+)
$f29$	8.67E+07 ± 2.25E+07	7.10E+07 ± 7.15E+07(−)	2.03E+08 ± 1.20E+08(+)	**3.85E+07 ± 8.14E+06(−)**	5.88E+08 ± 7.17E+07(+)
$f30$	9.40E+07 ± 2.85E+05	9.40E+07 ± 2.45E+05(≈)	9.41E+07 ± 2.11E+05(≈)	9.40E+07 ± 2.15E+05(≈)	9.33E+07 ± 3.28E+05(−)
$f31$	9.57E+11 ± 3.81E+10	9.29E+11 ± 9.06E+09(−)	**1.07E+09 ± 1.33E+09(−)**	9.25E+11 ± 8.70E+09(−)	3.74E+09 ± 4.07E+09(−)
$f32$	1.32E+03 ± 8.91E+01	**1.07E+03 ± 5.31E+01(−)**	1.17E+03 ± 7.56E+01(−)	1.87E+03 ± 1.72E+02(+)	8.69E+07 ± 1.95E+08(+)
$f33$	**2.13E+08 ± 1.10E+08**	7.97E+08 ± 2.78E+08(+)	1.90E+09 ± 6.75E+08(+)	2.17E + 08 ± 5.46E + 07(≈)	1.82E+09 ± 4.87E+08(+)
$f34$	**4.69E+07 ± 1.07E+07**	5.10E+09 ± 6.13E+09(+)	6.84E+09 ± 5.16E+09(+)	5.06E + 07 ± 2.16E + 07(≈)	1.87E+10 ± 5.40E+09(+)
$f35$	2.41E+07 ± 1.90E+06	7.83E+07 ± 7.03E+06(+)	1.44E+08 ± 3.64E+07(+)	**1.63E+06 ± 5.88E+04(−)**	2.30E+07 ± 1.66E+07(−)
+/−/≈	–	9/3/3	8/4/3	6/4/5	9/5/1



(a) $f1$ (b) $f2$ (c) $f3$

(d) $f21$ (e) $f22$ (f) $f23$

Fig. 6. Optimization process on separable functions.

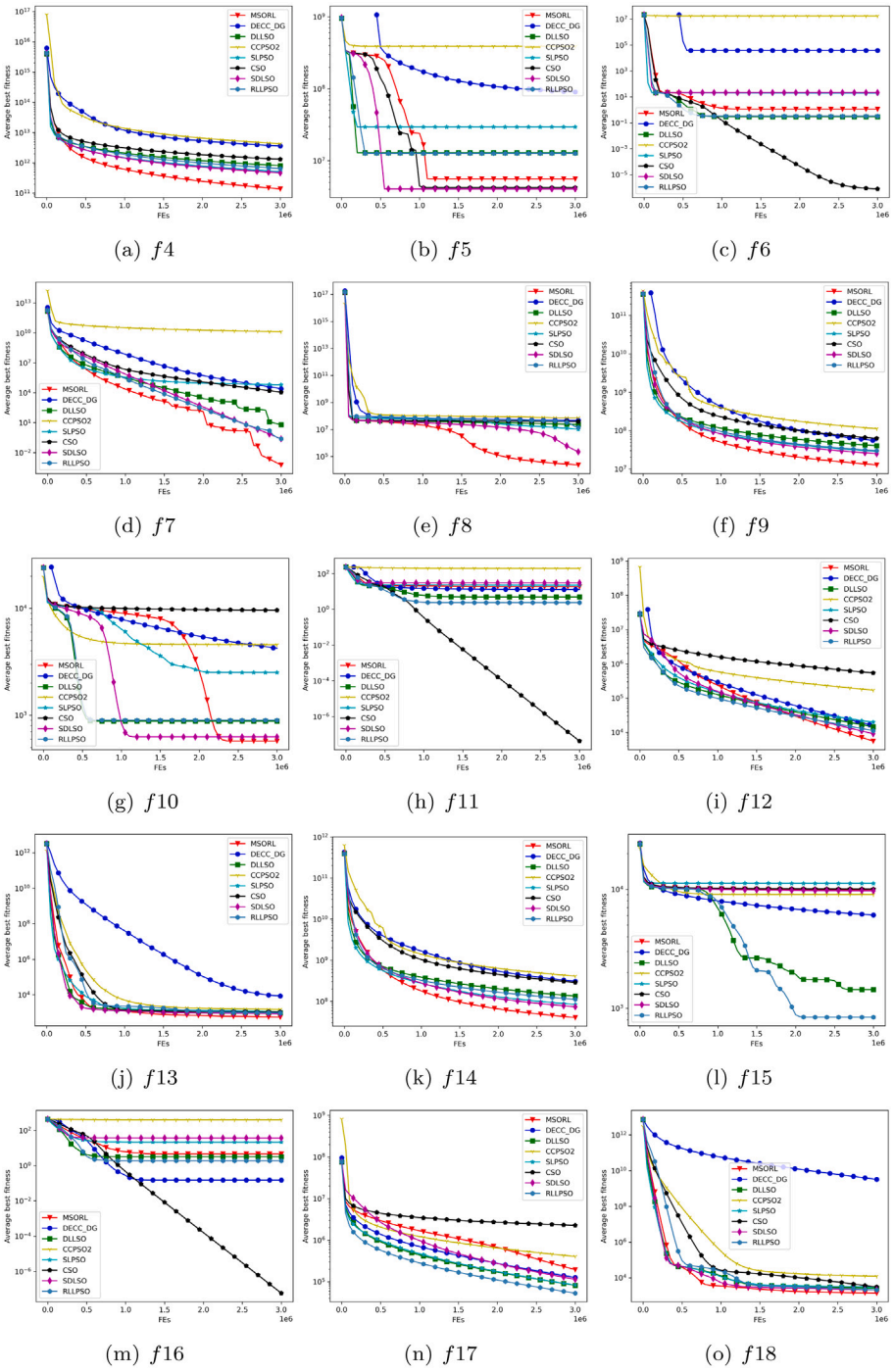


Fig. 7. Optimization process on CEC2010's partially separable functions.

more complex, the relative advantage of the MSORL becomes more pronounced. Therefore, the MSORL can be applied to solve complex LSGO problems, such as large-scale TSP and the neural architecture search problem.

4.4. Results and analysis of ablation experiments

To investigate the effectiveness of the proposed strategies, we conducted ablation experiments on CEC2010. To test the effectiveness of the tri-particle group structure, we eliminated the tri-particle group structure in the MSORL. Moreover, we designed a new algorithm named MSORL-W1, using a dual-particle group where suboptimal particles learn from the best particles. Additionally, the MSORL was modified by abandoning the RLG strategy, resulting in a new algorithm named MSORL-W2. Finally, the ATS was removed from the MSORL to create MSORL-W3. Table 4 presents the performance of the MSORL on CEC2010 in terms of the convergence accuracy.

Compared with MSORL-W1 and MSORL-W3, the MSORL performs better in terms of the convergence accuracy, achieving better results in 12 out of 20 test functions. Because the RLG strategy not only enhances the convergence efficiency but also reduces the diversity of the algorithm, the introduction of the ATS mechanism helps to ensure the diversity of the algorithm and to avoid the local optima. When

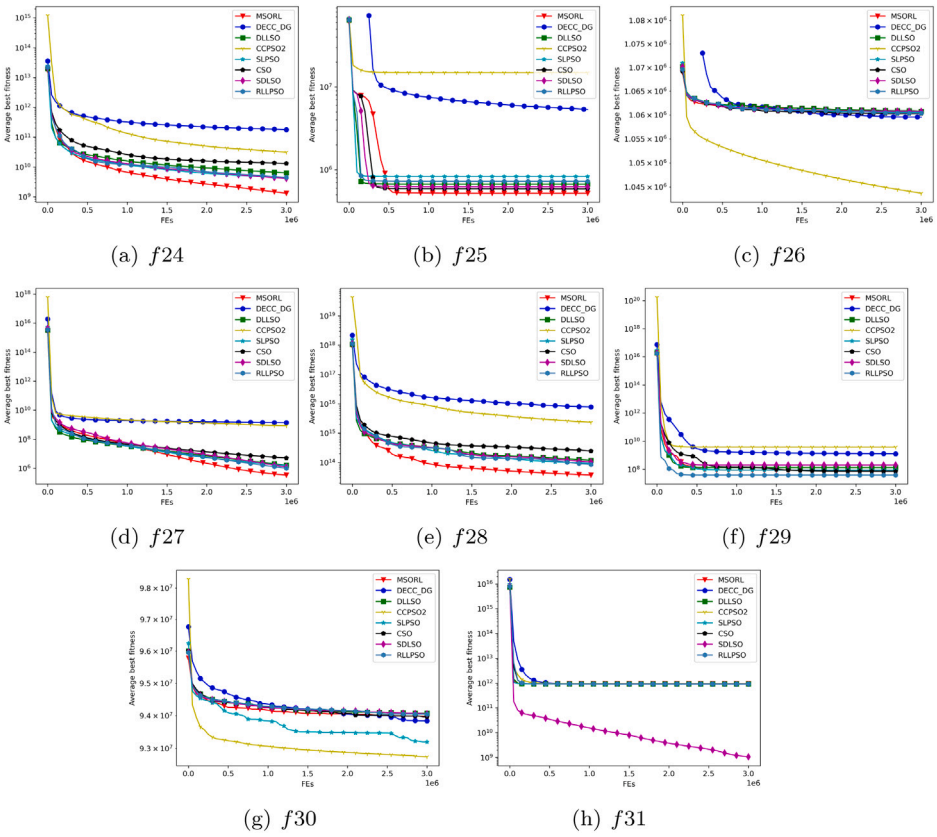


Fig. 8. Optimization process on CEC2013's partially separable functions.

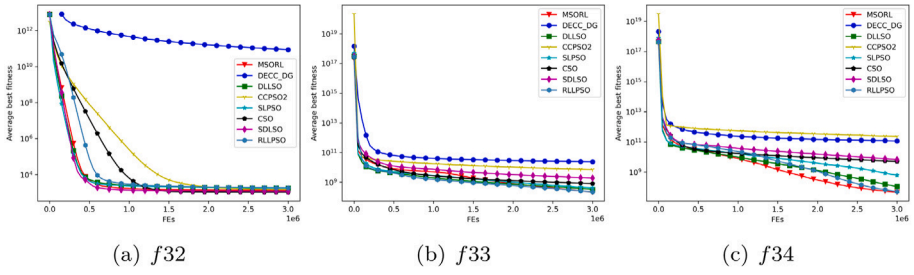


Fig. 9. Optimization process on overlapping functions.

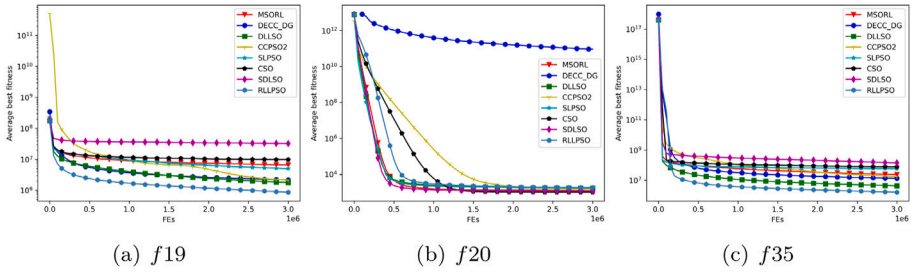


Fig. 10. Optimization process on non-separable functions.

the ATS mechanism is removed from MSORL-W3, particles start to become trapped in the local optima or converge prematurely in many test functions. However, when comparing the superior performance of MSORL-W3 to MSORL in certain test functions, we found that the advantage of MSORL-W3 over MSORL is not as significant as its disadvantage of trapping in the local optima.

In addition, compared with MSORL-W2, the main advantage of the MSORL in terms of the convergence accuracy is on unimodal functions ($f1$, $f4$, $f7$, $f9$, $f12$, $f14$, $f17$, and $f19$). This is mainly because although the RLG strategy increases the convergence efficiency of the MSORL, it weakens the diversity. Therefore, the performance of the MSORL on multimodal functions is slightly poor. However, the

Table 4
Comparison results of convergence accuracy in ablation experiments.

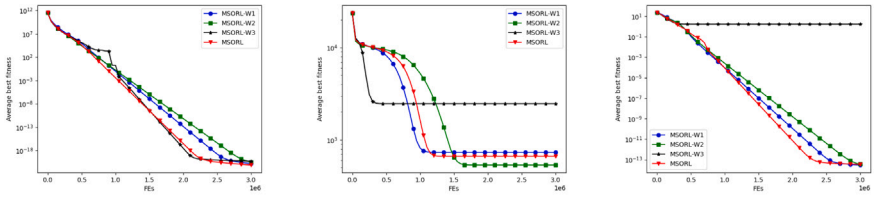
Functions		MSORL	MSORL-W1	MSORL-W2	MSORL-W3
$f1$	Mean	**7.90E−22**	1.67E−21	3.31E−21	5.39E−21
	Std	**1.10E−22**	1.32E−22	2.00E−22	7.69E−22
$f2$	Mean	6.66E+02	7.35E+02	**5.34E+02**	2.47E+03
	Std	2.74E+01	3.35E+01	**2.23E+01**	1.12E+02
$f3$	Mean	3.47E−14	**3.03E−14**	3.78E−14	1.69E+00
	Std	2.70E−15	1.77E−15	**1.71E−15**	1.16E−01
$f4$	Mean	1.38E+11	4.27E+11	1.49E+11	**1.34E+11**
	Std	**2.33E+10**	8.95E+10	2.56E+10	2.81E+10
$f5$	Mean	5.57E+06	7.83E+06	**4.82E+06**	1.97E+07
	Std	**1.68E+06**	2.42E+06	2.02E+06	3.27E+06
$f6$	Mean	1.15E+00	3.56E−09	**3.55E−09**	1.97E+01
	Std	4.69E−01	9.71E−13	**5.41E−13**	2.49E−02
$f7$	Mean	**5.89E−04**	1.49E+00	4.33E−02	2.68E+03
	Std	**3.47E−04**	8.68E−01	2.09E−02	7.50E+03
$f8$	Mean	2.38E+04	3.12E+06	**7.48E+02**	3.13E+04
	Std	1.05E+04	5.32E+05	**3.71E+02**	7.88E+03
$f9$	Mean	1.28E+07	3.18E+07	2.08E+07	**1.13E+07**
	Std	**1.06E+06**	2.40E+06	2.08E+06	1.10E+06
$f10$	Mean	5.73E+02	6.21E+02	**4.96E+02**	2.53E+03
	Std	3.05E+01	2.59E+01	**2.44E+01**	1.00E+02
$f11$	Mean	1.86E+01	4.25E−01	**5.31E−13**	3.19E+01
	Std	2.19E+00	5.83E−01	**6.64E−14**	1.24E+01
$f12$	Mean	5.56E+03	2.11E+04	4.04E+04	**8.19E+02**
	Std	**4.93E+02**	1.38E+03	2.44E+03	5.88E+02
$f13$	Mean	5.51E+02	6.75E+02	**5.17E+02**	5.87E+02
	Std	1.36E+02	1.28E+02	**1.02E+02**	1.44E+02
$f14$	Mean	4.01E+07	1.06E+08	7.88E+07	**3.55E+07**
	Std	2.90E+06	7.30E+06	4.02E+06	**2.25E+06**
$f15$	Mean	9.80E+03	9.71E+03	9.80E+03	**2.62E+03**
	Std	5.52E+01	**5.40E+01**	5.58E+01	1.12E+02
$f16$	Mean	4.49E+00	6.02E−01	**4.30E−13**	1.03E+02
	Std	3.73E+00	8.86E−01	**7.52E−14**	2.00E+01
$f17$	Mean	1.95E+05	1.68E+05	5.20E+05	**7.61E+03**
	Std	1.71E+04	6.62E+03	3.08E+04	**1.13E+03**
$f18$	Mean	1.37E+03	1.57E+03	**1.16E+03**	1.62E+03
	Std	3.31E+02	2.76E+02	**2.06E+02**	2.71E+02
$f19$	Mean	6.62E+06	5.02E+06	6.94E+06	**9.93E+05**
	Std	3.39E+05	**3.18E+05**	4.06E+05	3.89E+04
$f20$	Mean	1.23E+03	1.27E+03	**9.48E+02**	1.41E+03
	Std	7.36E+01	1.19E+02	**9.42E+00**	1.19E+02
	+/−/≈	–	12/1/7	8/3/9	12/2/6

overall performances of the MSORL and MSORL-W2 in terms of the convergence accuracy are similar, whereas the effectiveness of the RLG strategy can be proven in terms of the convergence speed performance.

Comparing the convergence accuracies of the MSORL and MSORL-W2 shows that after removing the RLG strategy, the performances of MSORL-W2 and the MSORL are similar, with approximately the same number of test functions having the best results. However, the main contribution of the RLG strategy to the algorithm is enhancing the convergence speed by selecting better learning exemplars for the particles. Therefore, to compare the convergence speed of MSORL, MSORL-W1, MSORL-W2, and MSORL-W3, the convergence processes on test functions are drawn. These are shown separately in Figs. 11, 13, and 12 according to the separability of the test functions.

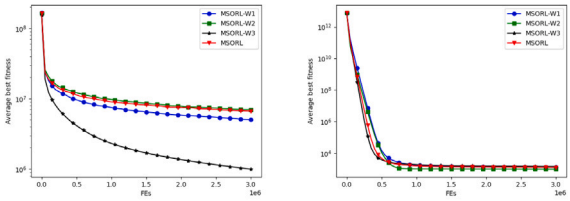
First, by observing the convergence process of MSORL (red line), MSORL-W1 (blue line) and MSORL-W2 (green line), we find that in most cases, the MSORL converges earlier than MSORL-W1 (i.e. $f1$, $f9$, $f12$, $f20$) and MSORL-W2 (i.e. $f1$, $f3$, $f5$, $f10$). Among the remaining functions also, the MSORL mostly converges faster. However, the higher convergence speed also causes prematurity in a few problems, such as in $f6$, $f11$, and $f16$. In summary, the MSORL is significantly faster than MSORL-W1 and MSORL-W2. Therefore, the tri-particle group structure and RLG strategy are considered effective in improving the convergence speed of the MSORL.

Furthermore, on functions $f2$, $f3$, $f5$, $f6$, $f10$, and $f16$, MSORL-W3, which lacks the ATS mechanism compared to MSORL-W2 and the MSORL, is frequently affected by the local optima during optimization.



(a) $f1$ (b) $f2$ (c) $f3$

Fig. 11. Convergence process on separable functions.



(a) $f19$ (b) $f20$

Fig. 12. Convergence process on non-separable functions.

It results in a low convergence speed or even premature problems. On functions $f2$, $f5$, $f7$, and $f10$, when all algorithms become trapped into the local optima, MSORL-W3 can only escape once and rapidly falls into a new local optimum after avoiding the local optima. Concurrently, MSORL-W2 and MSORL avoid the local optima multiple times and eventually obtain better convergence results. Therefore, MSORL-W3 converges rapidly in some problems such as $f12$, $f15$, $f17$, and $f19$ owing to the effect of the RLG strategy. Overall, it still performs at a significant disadvantage compared to the MSORL. This difference also proves the effectiveness of the ATS mechanism. It can adaptively determine the current state of the swarm and provide more diverse learning exemplars with more differences for optimization when the swarm falls into the local optima. It improves the diversity of the MSORL.

Based on ablation experiments, we designed multiple comparative algorithms to test the proposed strategies. Comparison of MSORL and MSORL-W1 in terms of the overall convergence accuracy shows the effectiveness of the tri-particle group structure. The high convergence speed and superior performances on unimodal functions prove the efficiency of the RLG strategy. The necessity of the ATS mechanism is proven by the poor performance of MSORL-W3 on multimodal functions. Comparing the MSORL with these comparative algorithms in terms of the convergence accuracy and speed proves the effectiveness of our proposed strategies.

5. Conclusion

In this study, we developed an MSORL, which combines RL and a multiswarm strategy. The multiswarm strategy of the MSORL allows different subswarms to explore different search space areas, enhancing the diversity of the algorithm. To select suitable updating strategies for different particles and balance the exploration and exploitation, the subswarms are organized into tri-particle groups. Furthermore, to enhance the convergence efficiency and overcome the uncertainty caused by random searches in traditional large-scale optimization algorithms, we designed an RLG strategy that guides particles from different subswarms to learn from the most promising exemplars. In addition, an ATS mechanism was adopted for the subswarms when they were trapped in the local optima, which helped to enhance the diversity of the subswarms by learning from other subswarms, allowing them to avoid the local optima.

The experimental results showed the efficiency of the MSORL in solving LSGO problems and the effectiveness of the proposed strategies. However, there is still scope for conducting studies in the future. In this study, considering the limits of the computational cost and structure of a PSO, we only adopted RL for discrete situations. Deep learning and neural network algorithms can be also used in LSGO problems. In addition, we can focus on further improving the convergence speed of the MSORL.

CRediT authorship contribution statement

Xujie Wang: Writing – original draft, Writing – review & editing. **Feng Wang:** Writing – original draft, Writing – review & editing, Conceptualization, Methodology. **Qi He:** Writing – review & editing. **Yinan Guo:** Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

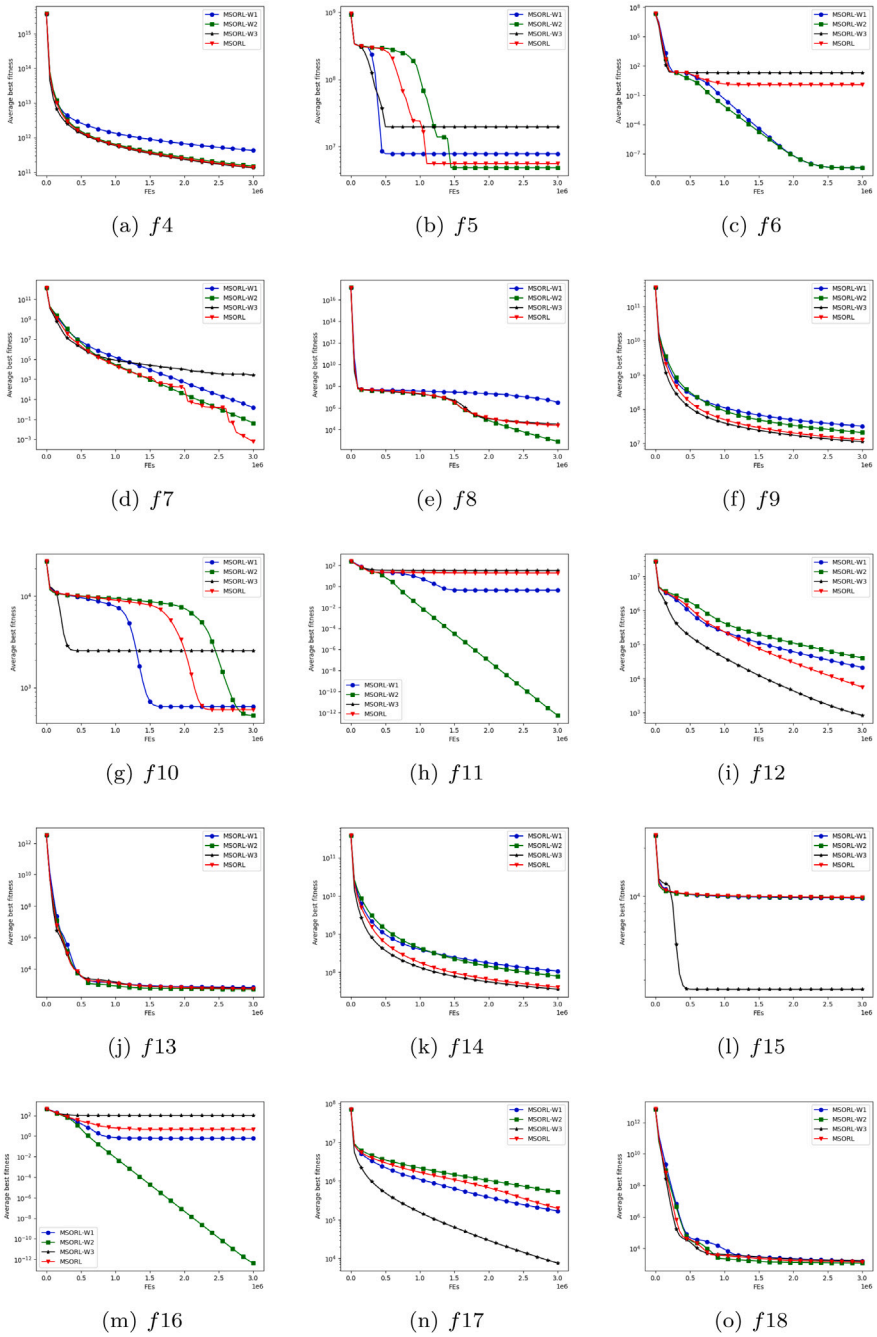
Acknowledgments

This work is supported by National Nature Science Foundation of China [Grant Nos. 62173258, 61773296].

Appendix. Experimental results on small-scale problems

We conducted experiments on CEC2013 (small-scale optimization problems) to compare the performance of the MSORL with those of eight state-of-the-art algorithms: DECC-DG [10], DLLSO [38], CCPSO2 [13], SL-PSO [20], CSO [19], SDLSO [7], RLLPSO [25], and LDE [47]. The maximum function evaluations (maxFE) are set at 600,000 with other parameters set to their optimal values.

The results in Table A.1 show that, MSORL outperforms DECC-DG, CCPSO2, SL-PSO, SDLSO, and LDE on most functions. However, DLLSO, CSO, and RLLPSO perform better than MSORL. This can be attributed to MSORL converging relatively slowly for training the reinforcement learning mechanism in the early stage of optimization, whereas other algorithms can converge earlier in simple small-scale problems.



(a) $f4$ (b) $f5$ (c) $f6$

(d) $f7$ (e) $f8$ (f) $f9$

(g) $f10$ (h) $f11$ (i) $f12$

(j) $f13$ (k) $f14$ (l) $f15$

(m) $f16$ (n) $f17$ (o) $f18$

Fig. 13. Convergence process on partially separable functions.

Table A.1
Comparison results on CEC2013 small-scale optimization problems.

Functions	MSORL Mean±Std	DECC-DG Mean±Std	DLLSO Mean±Std	CCPSO2 Mean±Std	SL-PSO Mean±Std
$f1$	0.0000E+00 ± 0.0000E+00	2.2737E−13 ± 0.0000E+00(≈)	0.0000E+00 ± 0.0000E+00(≈)	1.9630E−12 ± 4.1298E−13(≈)	1.8948E−13 ± 8.4737E−14(≈)
$f2$	**4.2000E+05 ± 9.0532E+04**	1.7340E+06 ± 6.2700E+05(+)	1.0572E+06 ± 2.8908E+05(+)	1.7692E+06 ± 8.9753E+05(+)	9.8175E+05 ± 3.2898E+05(+)
$f3$	1.5473E+06 ± 3.7809E+06	1.3350E+08 ± 1.7870E+08(+)	6.4630E+06 ± 7.3220E+06(+)	1.0374E+09 ± 8.3575E+08(+)	**1.6192E+07 ± 2.0754E+07(+)**
$f4$	2.7813E+03 ± 5.2537E+02	3.0019E+04 ± 3.9971E+03(+)	**2.8015E+02 ± 1.4953E+02(−)**	8.5717E+04 ± 2.4681E+04(+)	6.7459E+03 ± 2.0657E+03(+)
$f5$	0.0000E+00 ± 0.0000E+00	2.3874E−13 ± 3.4106E−14(≈)	4.5475E−14 ± 5.5695E−14(≈)	3.0278E−09 ± 8.9869E−09(≈)	1.8190E−13 ± 5.5695E−14(≈)
$f6$	4.3447E+01 ± 3.4371E−14	4.7554E+01 ± 1.2203E+01(+)	**4.5286E+01 ± 4.1306E−01(+)**	4.4891E+01 ± 8.3427E+00(+)	4.3531E+01 ± 2.1129E−01(+)
$f7$	**2.8146E+00 ± 1.5977E+00**	6.5392E+01 ± 1.1715E+01(+)	4.2580E+00 ± 2.0805E+00(+)	1.2827E+02 ± 1.9116E+01(+)	6.0236E+00 ± 3.4471E+00(+)
$f8$	2.1123E+01 ± 4.4858E−02	2.1152E+01 ± 3.4728E−02(≈)	2.1121E+01 ± 2.4458E−02(≈)	2.1135E+01 ± 2.8324E−02(≈)	**2.1117E+01 ± 2.8823E−02**(≈)
$f9$	**1.3418E+01 ± 3.7609E+00**	5.9639E+01 ± 2.6123E+00(+)	1.4740E+01 ± 2.5466E+00(+)	5.4691E+01 ± 2.4322E+00(+)	1.6579E+01 ± 2.7378E+00(+)
$f10$	**8.3848E−02 ± 3.3622E−02**	2.2162E−01 ± 1.4057E−01(+)	3.5622E−01 ± 1.9245E−01(+)	1.2853E−01 ± 5.3517E−02(≈)	2.9552E−01 ± 2.3595E−01(+)
$f11$	2.2552E+00 ± 1.8691E+00	3.0844E+00 ± 1.8409E+00(+)	6.6331E+00 ± 3.0215E+00(+)	**6.7887E−02 ± 2.4897E−01(−)**	1.5821E+01 ± 3.6445E+00(+)
$f12$	2.9195E+02 ± 1.1969E+01	2.0733E+02 ± 2.2640E+01(−)	1.0812E+01 ± 2.8460E+00(−)	**4.2751E+02 ± 8.7196E+01(+)**	2.4892E+02 ± 1.1256E+02(−)
$f13$	2.8264E+02 ± 5.3032E+01	2.7604E+02 ± 2.3807E+01(−)	2.1598E+01 ± 1.3015E+01(−)	**5.0715E+02 ± 6.9766E+01(+)**	2.5642E+02 ± 9.8291E+01(−)
$f14$	1.0821E+04 ± 5.3207E+02	6.2871E+01 ± 1.5197E+01(−)	1.1191E+02 ± 7.8230E+01(−)	**4.6578E+00 ± 2.3418E+00(−)**	3.4414E+02 ± 2.2024E+02(−)
$f15$	1.2158E+04 ± 4.8385E+02	**1.1490E+04 ± 5.5938E+02(−)**	1.0927E+03 ± 2.8467E+02(−)	7.5135E+03 ± 7.9346E+02(−)	1.9144E+03 ± 5.9338E+02(−)
$f16$	3.3966E+00 ± 2.5055E−01	**2.9899E+00 ± 3.1621E−01(−)**	3.3714E+00 ± 1.8650E−01(≈)	2.9952E+00 ± 3.2829E−01(−)	3.2557E+00 ± 2.9814E−01(−)
$f17$	3.3706E+02 ± 1.4468E+01	5.4175E+01 ± 8.1296E−01(−)	1.9939E+02 ± 1.9441E+01(−)	**5.1309E+01 ± 1.6533E−01(−)**	2.1892E+02 ± 2.1690E+01(−)
$f18$	5.1356E+01 ± 1.1309E+00	5.3402E+01 ± 1.8071E+00(+)	5.6791E+01 ± 2.3385E+00(+)	**5.0303E+01 ± 2.8648E−01(−)**	6.6247E+01 ± 3.6212E+00(+)
$f19$	2.2523E+01 ± 3.7005E+00	2.0850E+00 ± 1.6661E−01(−)	5.7808E+00 ± 5.9778E−01(−)	**1.2896E+00 ± 1.7189E−01(−)**	6.2883E+00 ± 7.6987E−01(−)
$f20$	1.8573E+01 ± 4.1927E−01	2.0915E+01 ± 3.7241E−01(+)	**1.7740E+01 ± 4.6504E−01(−)**	2.1092E+01 ± 6.2211E−01(+)	2.0713E+01 ± 4.3709E−01(+)
$f21$	3.1667E+02 ± 3.7268E+01	3.0667E+02 ± 2.4944E+01(−)	3.1000E+02 ± 3.0000E+01(−)	**2.6713E+02 ± 5.9159E+01(−)**	3.0667E+02 ± 2.4944E+01(−)
$f22$	7.9002E+03 ± 1.1430E+03	1.3055E+03 ± 1.1996E+03(−)	1.3752E+02 ± 6.0447E+01(−)	**8.8473E+01 ± 8.4674E+01(−)**	3.4597E+02 ± 1.8462E+02(−)
$f23$	3.5885E+03 ± 3.2964E+03	**1.2433E+04 ± 5.3881E+02(+)**	9.5473E+02 ± 3.4818E+02(−)	9.8590E+03 ± 1.0370E+03(+)	2.2255E+03 ± 6.3260E+02(−)
$f24$	2.3037E+02 ± 1.1708E+01	**2.6512E+02 ± 2.5227E+01(+)**	2.2133E+02 ± 1.2395E+01(−)	3.5154E+02 ± 7.2311E+00(+)	2.3315E+02 ± 1.2926E+01(+)
$f25$	2.8378E+02 ± 7.5303E+00	3.7513E+02 ± 1.0786E+01(+)	**2.8361E+02 ± 6.4320E+00(−)**	3.9034E+02 ± 6.7682E+00(+)	2.9290E+02 ± 6.0594E+00(+)
$f26$	3.0423E+02 ± 5.0478E+00	2.7367E+02 ± 1.0812E+02(−)	3.0473E+02 ± 1.1093E+01(+)	2.0268E+02 ± 1.2973E+01(−)	**3.1637E+02 ± 1.4506E+01(+)**
$f27$	6.2955E+02 ± 1.6172E+02	**1.6844E+03 ± 1.4034E+02(+)**	5.7415E+02 ± 1.3380E+02(−)	1.7556E+03 ± 6.0568E+01(+)	6.4900E+02 ± 1.6181E+02(+)
$f28$	4.0000E+02 ± 2.0495E−13	5.0138E+02 ± 5.4592E+02(+)	4.0000E+02 ± 2.2307E−13(≈)	4.0000E+02 ± 1.4710E−11(≈)	**5.9835E+02 ± 7.4215E+02(+)**
+/−/≈	–	15/10/3	9/14/5	13/10/5	15/10/3

Functions	MSORL Mean±Std	CSO Mean±Std	SDLSO Mean±Std	RLLPSO Mean±Std	LDE Mean±Std
$f1$	0.0000E+00 ± 0.0000E+00	0.0000E+00 ± 0.0000E+00(≈)	0.0000E+00 ± 0.0000E+00(≈)	0.0000E+00 ± 0.0000E+00(≈)	9.6113E−09 ± 3.1463E−10(≈)
$f2$	**4.2000E+05 ± 9.0532E+04**	1.8352E+06 ± 3.9047E+05(+)	4.5835E+05 ± 1.0907E+05(+)	4.3406E+05 ± 1.0991E+05(+)	8.8228E+05 ± 2.7523E+05(+)
$f3$	1.5473E+06 ± 3.7809E+06	1.9698E+06 ± 2.8508E+06(+)	1.1036E+06 ± 2.6910E+06(−)	7.4720E+05 ± 1.4303E+06(−)	**1.3990E+03 ± 7.0836E+02(−)**
$f4$	2.7813E+03 ± 5.2537E+02	6.0378E+03 ± 1.0220E+03(+)	9.5788E+02 ± 3.5014E+02(−)	4.9718E+02 ± 2.2784E+02(−)	9.3323E+03 ± 2.2575E+03(+)
$f5$	0.0000E+00 ± 0.0000E+00	1.1748E−13 ± 2.0407E−14(≈)	0.0000E+00 ± 0.0000E+00(≈)	4.5475E−14 ± 5.5695E−14(≈)	9.7653E−09 ± 2.2626E−10(≈)
$f6$	4.3447E+01 ± 3.4371E−14	4.5951E+01 ± 3.4478E−01(+)	**3.7029E+01 ± 4.9620E−01(−)**	4.4501E+01 ± 4.1600E−01(+)	4.3854E+01 ± 5.5791E−01(+)
$f7$	**2.8146E+00 ± 1.5977E+00**	3.9203E+00 ± 1.5498E+00(+)	4.3577E+00 ± 2.1094E+00(+)	2.8314E + 00 ± 1.2207E + 00(≈)	1.1724E+01 ± 4.3982E+00(+)
$f8$	2.1123E+01 ± 4.4858E−02	2.1119E + 01 ± 4.0157E − 02(≈)	2.1123E+01 ± 3.8404E−02(≈)	2.1134E+01 ± 3.6244E−02(≈)	2.1133E+01 ± 4.0441E−02(≈)
$f9$	**1.3418E+01 ± 3.7609E+00**	1.5490E+01 ± 3.1138E+00(+)	1.8578E+01 ± 1.4729E+01(+)	1.4689E+01 ± 3.2945E+00(+)	6.1058E+01 ± 1.8458E+00(+)
$f10$	**8.3848E−02 ± 3.3622E−02**	3.3792E−01 ± 2.0404E−01(+)	1.2415E−01 ± 6.4325E−02(≈)	1.0832E − 01 ± 4.0906E − 02(≈)	1.5809E−01 ± 9.3221E−02(+)
$f11$	2.2552E+00 ± 1.8691E+00	1.7246E+00 ± 1.1178E+00(−)	2.8191E+00 ± 1.8181E+00(+)	4.8090E+00 ± 1.6862E+00(+)	1.5666E+02 ± 6.3355E+00(+)
$f12$	2.9195E+02 ± 1.1969E+01	2.9776E+02 ± 1.0135E+01(+)	1.8866E+01 ± 5.2699E+01(−)	**8.5899E+00 ± 3.0879E+00(−)**	2.7317E+02 ± 2.0984E+01(−)
$f13$	2.8264E+02 ± 5.3032E+01	2.9316E+02 ± 1.1385E+01(+)	2.4442E+01 ± 4.8460E+01(−)	**1.5065E+01 ± 8.1779E+00(−)**	2.8932E+02 ± 2.1625E+01(+)
$f14$	1.0821E+04 ± 5.3207E+02	3.3198E+03 ± 1.1205E+03(−)	1.1722E+04 ± 6.7853E+02(+)	1.1397E+02 ± 6.5091E+01(−)	7.8542E+03 ± 1.8586E+02(−)
$f15$	1.2158E+04 ± 4.8385E+02	**7.2124E+02 ± 2.5794E+02(−)**	1.2370E+04 ± 4.1210E+02(+)	1.2157E+03 ± 3.5313E+02(−)	1.3367E+04 ± 3.0761E+02(+)
$f16$	3.3966E+00 ± 2.5055E−01	3.3315E+00 ± 2.9895E−01(−)	3.4005E+00 ± 2.3506E−01(≈)	3.2175E+00 ± 3.1305E−01(−)	3.2119E+00 ± 3.4299E−01(−)
$f17$	3.3706E+02 ± 1.4468E+01	3.3086E+02 ± 1.5934E+01(−)	3.0160E+02 ± 1.8073E+01(−)	2.5412E+02 ± 3.2074E+01(−)	2.1691E+02 ± 6.5588E+00(−)
$f18$	5.1356E+01 ± 1.1309E+00	5.1155E+01 ± 7.8075E−01(−)	5.2824E+01 ± 1.6518E+00(+)	5.4825E+01 ± 1.9001E+00(+)	3.8202E+02 ± 1.1479E+01(+)
$f19$	2.2523E+01 ± 3.7005E+00	5.7144E+00 ± 5.1756E−01(−)	5.4042E+00 ± 5.2135E−01(−)	5.7311E+00 ± 4.9171E−01(−)	2.0066E+01 ± 1.4835E+00(−)
$f20$	1.8573E+01 ± 4.1927E−01	1.9108E+01 ± 3.7131E−01(+)	1.7751E+01 ± 4.7944E−01(−)	1.7981E+01 ± 4.7478E−01(−)	2.1478E+01 ± 4.0388E−01(+)
$f21$	3.1667E+02 ± 3.7268E+01	3.0000E+02 ± 7.4694E−13(−)	3.1333E+02 ± 3.3993E+01(−)	3.2000E+02 ± 4.0000E+01(+)	9.0398E+02 ± 3.0495E+02(+)
$f22$	7.9002E+03 ± 1.1430E+03	1.1335E+03 ± 6.3461E+02(−)	1.0725E+04 ± 9.4815E+02(+)	1.0651E+02 ± 4.8970E+01(−)	8.0157E+03 ± 3.3936E+02(+)
$f23$	3.5885E+03 ± 3.2964E+03	**4.2791E+02 ± 2.2376E+02(−)**	1.1993E+04 ± 5.9872E+02(+)	9.4417E+02 ± 3.1703E+02(−)	1.3603E+04 ± 3.7943E+02(+)
$f24$	2.3037E+02 ± 1.1708E+01	**2.0747E+02 ± 9.2581E+00(−)**	2.8927E+02 ± 6.9887E+01(+)	2.1887E+02 ± 1.4015E+01(−)	2.1892E+02 ± 4.0966E+00(−)
$f25$	2.8378E+02 ± 7.5303E+00	2.8878E+02 ± 7.6535E+00(+)	3.8105E+02 ± 3.9239E+00(+)	2.8607E+02 ± 6.9039E+00(+)	3.5873E+02 ± 3.2384E+01(+)
$f26$	3.0423E+02 ± 5.0478E+00	3.0398E+02 ± 4.3416E+00(−)	3.1319E+02 ± 1.0300E+01(+)	2.9955E+02 ± 1.8814E+01(−)	**2.0014E+02 ± 5.2661E−02(−)**
$f27$	6.2955E+02 ± 1.6172E+02	**4.3597E+02 ± 1.4922E+02(−)**	7.7811E+02 ± 3.6263E+02(+)	4.8022E+02 ± 1.5280E+02(−)	5.8192E+02 ± 7.5170E+01(−)
$f28$	4.0000E+02 ± 2.0495E−13	4.0000E+02 ± 2.3802E−13(≈)	4.0000E + 02 ± 1.3371E − 13(≈)	4.0000E+02 ± 2.0963E−13(≈)	**4.0000E+02 ± 1.7975E−14**(≈)
+/−/≈	–	11/13/4	13/9/6	7/15/6	15/9/4

References

[1] Yanwei Pang, Lei Zhang, Zhengkai Liu, Nenghai Yu, Houqiang Li, Xiao-Ping Zhang, Guang-Bin Huang, Neighborhood preserving projections (npp): A novel linear dimension reduction method, in: Advances in Intelligent Computing, Springer Berlin Heidelberg, 2005, pp. 117–125.

[2] Nandar Lynn, Ponnuthurai Nagaratnam Suganthan, Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24.

[3] Haiyan Liu, Yuping Wang, Ninglei Fan, A hybrid deep grouping algorithm for large scale global optimization, IEEE Trans. Evol. Comput. 24 (6) (2020) 1112–1124.

[4] Xin Zhang, Ke-Jing Du, Zhi-Hui Zhan, Sam Kwong, Tian-Long Gu, Jun Zhang, Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties, IEEE Trans. Cybern. 50 (10) (2020) 4454–4468.

[5] X.Y. Zhang, J. Zhang, Yue Jiao Gong, Zhi Hui Zhan, W.N. Chen, Y. Li, Kuhn-munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks, IEEE Trans. Evol. Comput. 20 (5) (2016) 695–710.

[6] Yusuf Nasir, Oleg Volkov, Louis J. Durlofsky, A two-stage optimization strategy for large-scale oil field development, Opt. Eng. (2021) 1–35.

[7] Qiang Yang, Wei-Neng Chen, Tianlong Gu, Hu Jin, Wentao Mao, Jun Zhang, An adaptive stochastic dominant learning swarm optimizer for high-dimensional optimization, IEEE Trans. Cybern. 52 (3) (2022) 1960–1976.

[8] Mitchell A. Potter, Kenneth A. De Jong, A cooperative coevolutionary approach to function optimization, in: Parallel Problem Solving from Nature, PPSN III, Springer Berlin Heidelberg, 1994, pp. 249–257.

[9] Van Den Bergh Frans, Andries P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.

[10] M.N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 378–393.

[11] Yi Mei, Mohammad Nabi Omidvar, Xiaodong Li, Xin Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. 42 (2) (2016) 1–24.

[12] Xiaodong Li, Xin Yao, Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 1546–1553.

[13] Xiaodong Li, Xin Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2012) 210–224.

[14] Tapabrata Ray, Xin Yao, A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 983–989.

[15] Qiang Yang, Wei-Neng Chen, Tianlong Gu, Huaxiang Zhang, Jeremiah D. Deng, Yun Li, Jun Zhang, Segment-based predominant learning swarm optimizer for large-scale optimization, IEEE Trans. Cybern. 47 (9) (2017) 2896–2910.

[16] Sibghat Ullah, Duc Anh Nguyen, Hao Wang, Stefan Menzel, Bernhard Sendhoff, Thomas Back, Exploring dimensionality reduction techniques for efficient surrogate-assisted optimization, in: 2020 IEEE Symposium Series on Computational Intelligence, SSCI, 2020, pp. 2965–2974.

[17] Shi Cheng, Yuhui Shi, Quande Qin, Dynamical exploitation space reduction in particle swarm optimization for solving large scale problems, in: 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.

[18] Hong-Rui Wang, Chun-Hua Chen, Yun Li, Jun Zhang, Progressive sampling surrogate-assisted particle swarm optimization for large-scale expensive optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2022, pp. 40–48.

[19] Ran Cheng, Yaochu Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybern. 45 (2) (2015) 191–204.

[20] Ran Cheng, Yaochu Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inform. Sci. 291 (2015) 43–60.

[21] Hanbo Deng, Lizhi Peng, Haibo Zhang, Bo Yang, Zhenxiang Chen, Ranking-based biased learning swarm optimizer for large-scale optimization, Inform. Sci. 493 (2019) 120–137.

[22] Han, Huang, Liang, Lv, Shujin, Ye, Zhifeng, Hao, Particle swarm optimization with convergence speed controller for large-scale numerical optimization, Soft Comput. Fusion Found. Methodol. Appl. (2018) 4421–4437.

[23] S.Z. Zhao, J.J. Liang, P.N. Suganthan, M.F. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization, in: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 2008, pp. 3845–3852.

[24] Qiang Yang, Wei-Neng Chen, Tianlong Gu, Huaxiang Zhang, Huaqiang Yuan, Sam Kwong, Jun Zhang, A distributed swarm optimizer with adaptive communication for large-scale optimization, IEEE Trans. Cybern. 50 (7) (2019) 3393–3408.

[25] Feng Wang, Xujie Wang, Shilei Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, Inform. Sci. 602 (2022) 298–312.

[26] Madalina M. Drugan, Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms, Swarm Evol. Comput. 44 (2019) 228–246.

[27] Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, Stefan Schaal, Learning objective functions for manipulation, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 1331–1336.

[28] Chia-Feng Juang, Trong Bac Bui, Reinforcement neural fuzzy surrogate-assisted multiobjective evolutionary fuzzy systems with robot learning control application, IEEE Trans. Fuzzy Syst. 28 (3) (2020) 434–446.

[29] Amir Seyyedabbasi, Royal Aliyev, Farzad Kiani, Murat Ugur Gulle, Hasan Basyildiz, Mohammed Ahmed Shah, Hybrid algorithms based on combining reinforcement learning and metaheuristic methods to solve global optimization problems, Knowl.-Based Syst. 223 (2021) 107044.

[30] Giorgos Karafotias, Agoston Endre Eiben, Mark Hoogendoorn, Generic Parameter Control with Reinforcement Learning, GECCO '14, Association for Computing Machinery, 2014, pp. 1319–1326.

[31] Y. Xu, D. Pi, A reinforcement learning-based communication topology in particle swarm optimization, Neural Comput. Appl. 32 (14) (2020) 10007–10032.

[32] Thomas Philip Runarsson, Learning heuristic policies–a reinforcement learning problem, in: Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5, Springer, 2011, pp. 423–432.

[33] Minyang Chen, Wei Du, Yang Tang, Yaochu Jin, Gary G. Yen, A decomposition method for both additively and non-additively separable problems, IEEE Trans. Evol. Comput. (2022) 1–16.

[34] Weikuan Jia, Meili Sun, Jian Lian, Sujuan Hou, Feature dimensionality reduction: a review, Complex Intell. Syst. 8 (3) (2022) 2663–2693.

[35] Jun-Rong Jian, Zong-Gan Chen, Zhi-Hui Zhan, Jun Zhang, Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization, IEEE Trans. Evol. Comput. 25 (4) (2021) 779–793.

[36] Dongyang Li, Weian Guo, Alexander Lerch, Yongmei Li, Lei Wang, Qidi Wu, An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization, Swarm Evol. Comput. 60 (2021) 100789.

[37] H. Wang, M. Liang, C. Sun, G. Zhang, L. Xie, Multiple-strategy learning particle swarm optimization for large-scale optimization problems, Complex Intell. Syst. (2020) 1–16.

[38] Qiang Yang, Wei-Neng Chen, Jeremiah Da Deng, Yun Li, Tianlong Gu, Jun Zhang, A level-based learning swarm optimizer for large-scale optimization, IEEE Trans. Evol. Comput. 22 (4) (2018) 578–594.

[39] Zi-Jia Wang, Zhi-Hui Zhan, Wei-Jie Yu, Ying Lin, Jie Zhang, Tian-Long Gu, Jun Zhang, Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling, IEEE Trans. Cybern. 50 (6) (2020) 2715–2729.

[40] Zi-Jia Wang, Zhi-Hui Zhan, Sam Kwong, Hu Jin, Jun Zhang, Adaptive granularity learning distributed particle swarm optimization for large-scale optimization, IEEE Trans. Cybern. 51 (3) (2021) 1175–1188.

[41] Ling Wang, Zixiao Pan, Jingjing Wang, A review of reinforcement learning based intelligent optimization for manufacturing scheduling, Complex Syst. Model. Simul. 1 (4) (2021) 257–270.

[42] Hussein Samma, Chee Peng Lim, Junita Mohamad Saleh, A new reinforcement learning-based memetic particle swarm optimizer, Appl. Soft Comput. 43 (2016) 276–297.

[43] Yaxian Liu, Hui Lu, Shi Cheng, Yuhui Shi, An adaptive online parameter control algorithm for particle swarm optimization based on reinforcement learning, in: 2019 IEEE Congress on Evolutionary Computation, CEC, 2019, pp. 815–822.

[44] Ke Tang, Xiaodong Li, P.N. Suganthan, Zhenyu Yang, Thomas Weise, Benchmark functions for the cec'2010 special session and competition on large-scale global optimization, Nat. Inspir. Comput. Appl. Lab. (2010) 1–18.

[45] Xiaodong Li, Ke Tang, Mohammad N. Omidvar, Zhenyu Yang, Kai Qin, Hefei China, Benchmark functions for the cec 2013 special session and competition on large-scale global optimization, gene 7 (33) (2013) 1–23.

[46] Tuo Zhang, Handing Wang, Bo Yuan, Yaochu Jin, Xin Yao, Surrogate-assisted evolutionary q-learning for black-box dynamic time-linkage optimization problems, IEEE Trans. Evol. Comput. (2022) 1–15.

[47] Jianyong Sun, Xin Liu, Thomas Back, Zongben Xu, Learning adaptive differential evolution algorithm from optimization experiences by policy gradient, IEEE Trans. Evol. Comput. 25 (4) (2021) 666–680.