Expert Systems With Applications 246 (2024) 123150

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A reinforcement learning-based neighborhood search operator for multi-modal optimization and its applications

Jiale Hong, Bo Shen *, Anqi Pan

College of Information Science and Technology, Donghua University, Shanghai, 201620, China
Engineering Research Center of Digitalized Textile and Fashion Technology, Ministry of Education, Shanghai, 201620, China

ARTICLE INFO

Keywords:
Multi-modal optimization problem
Niching methods
Reinforcement learning
Inverse kinematics

ABSTRACT

In this paper, a reinforcement learning-based neighborhood search operator (RLNS) is proposed for multi-modal optimization problems where the main novelties lie in the reinforcement learning-based neighborhood range selection strategy, the neighborhood subpopulation generation strategy and the local vector encirclement model. The reinforcement learning-based neighborhood range selection strategy is proposed to dynamically adjust the subpopulation size to address the issue of too many parameters to be adjusted in the multi-modal optimization algorithm based on the niching methods, while the neighborhood subpopulation generation strategy and the local vector encirclement model are designed with the hope of enhancing the individual's ability to local exploitation to obtain more accurate solutions. To verify the effectiveness of the proposed RLNS, SSA-RLNS, PSO-RLNS and EO-RLNS are proposed by integrating the proposed RLNS with the existing sparrow search algorithm, particle swarm optimization and equilibrium optimizer. The performances of the proposed SSA-RLNS, PSO-RLNS, EO-RLNS and existing multi-modal optimization algorithms are tested in CEC2015 multi-niche benchmark functions. The experimental results show that the SSA-RLNS, PSO-RLNS and EO-RLNS could locate multiple global optimal solutions with satisfactory accuracy, which illustrate that the proposed RLNS could be successfully used to deal with multi-modal optimization problems by integrating with common population-based optimization algorithms. Finally, the SSA-RLNS is successfully applied in the inverse kinematics of robot manipulator.

1. Introduction

In the real-world engineering field, there are many optimization problems, such as robot manipulator (RM) kinematics (Dereli & Koker, 2020), flow shop scheduling (Mohammadi & Moaddabi, 2021), protein structure prediction (Song et al., 2018), and so on Hernandez-Barragan et al. (2021), have multiple global and local optimal solutions which are called multi-modal optimization problems (MMOPs) (El-Sherbiny et al., 2018; Wang et al., 2012).

Recently, several population-based optimization algorithms, such as Particle Swarm Optimization (PSO) (Eberhart & Kenned, 1995), Sparrow Search Algorithm (SSA) (Xue & Shen, 2020), Equilibrium Optimizer (EO) (Faramarzi et al., 2020), Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016), have been proposed for solving complex nonlinear (Duc et al., 2018; Precup et al., 2021b), multi-objective (Pan et al., 2022), constrained engineering optimization problems (Wang et al., 2019), which have demonstrated excellent performance. It is worth mentioning that some advanced technologies have also been reported for optimization and decision-making problems (Precup et al., 2021a), such as Manifold optimization (Singh & Shukla, 2022), analytical hierarchy process (Lin, 2022), neural networks and reinforcement learning (Tan et al., 2014; Zamfirache et al., 2023). Algorithms based on random optimization methods suffer from limitations in handling MMOPs, as they can only locate a single global optimal solution (Jiang et al., 2022). Therefore, to improve the performance of these population-based algorithms in solving complex MMOPs, researchers have explored the combination of multi-modal optimization techniques with population-based algorithms (Li et al., 2017).

The niching methods are a primary approach for addressing MMOPs, which utilizes some specific techniques to partition the population into multiple subpopulations to fully search the entire search space, and achieve the balance of the global exploration and local exploitation ability of the algorithm (Farshi & Orujpour, 2021; Luo & Gu, 2016). Various niching methods have been developed to enhance population-based optimization algorithms for MMOPs, which include derating (Zhang et al., 2006), fitness sharing (Rim et al., 2018), neighborhood mutation (Das et al., 2009; Epitropakis et al., 2011), speciation (Li et al., 2002), clustering (Gao et al., 2014), and so on.

* Corresponding author at: College of Information Science and Technology, Donghua University, Shanghai, 201620, China.
E-mail addresses: hongjiale@mail.dhu.edu.cn (J. Hong), bo.shen@dhu.edu.cn (B. Shen), pananqi@dhu.edu.cn (A. Pan).

https://doi.org/10.1016/j.eswa.2024.123150
Received 29 March 2023; Received in revised form 8 September 2023; Accepted 2 January 2024
Available online 5 January 2024

By utilizing niching methods, the algorithms can effectively handle multiple peaks in the search space, leading to better population diversity. In Qu et al. (2012), a neighborhood mutation was proposed for MMOPs, which was integrated with differential evolution to effectively search for the multiple optima. A new multi-modal optimization algorithm based on the firefly algorithm was presented in Nekouie and Yaghoobi (2016). The algorithm enhanced the diversity of the population and the quality of the optima in the firefly algorithm by dividing the population into multiple subpopulations. In Rim et al. (2018), a niche chaos optimization algorithm was proposed that employed the distance between multiple subpopulations as an index to evaluate the diversity of communities to improve the convergence speed and accuracy of the algorithm. An improved differential evolution is proposed in Lin et al. (2021) that combines the differential evolution with the nearest-better clustering method for solving MMOPs, which effectively balances the trade-off between exploration and exploitation.

Despite the extensive research on niching methods have been investigated to improve the performance of the population-based algorithms for MMOPs, these methods exhibit significant challenges, including too much niching parameters are required to be tuned and the difficulty in obtaining solutions with high precision (Orujpour et al., 2020; Tanabe & Ishibuchi, 2019; Yoo, 2018). Therefore, it is of great significance to design an operator for MMOPs, which not only could adjust parameters adaptively but also effectively balances the trade-off between exploration and exploitation. This constitutes the main motivation of this paper. The primary contributions of this paper are as follows: (1) a reinforcement learning-based neighborhood search operator (RLNS) is proposed for MMOPs where the main novelties lie in the reinforcement learning-based neighborhood range selection strategy, the neighborhood subpopulation generation strategy and the local vector encirclement model; (2) three multi-modal optimization algorithms, SSA-RLNS, PSO-RLNS and EO-RLNS, are proposed by integrating the proposed RLNS with the existing SSA, PSO and EO; (3) The proposed SSA-RLNS is successfully applied to the inverse kinematics (IK) research of the RM.

The structure of this paper is organized as follows. The preliminaries and related work are detailed in Section 2. Section 3 describes the proposed RLNS, SSA-RLNS, PSO-RLNS and EO-RLNS. The parameter setting, benchmark functions and experimental results are presented in Section 4. In Section 5, the application of proposed SSA-RLNS in the IK of RM is implemented. Finally, the paper is concluded in Section 6.

2. Preliminaries and related work

In this section, the basic principle of the Q-learning is first briefly introduced, and then some multi-modal optimization algorithms based on niching methods are presented.

2.1. Q-learning

Reinforcement learning is a type of machine learning that focuses on how an agent can maximize rewards by interacting with an environment. Q-learning is a reinforcement learning algorithm that determines the best actions for each state by updating the Q-values of these actions. In Q-learning, the agent selects actions based on the current state and the best action stored in the Q-table. As the agent interacts more with the environment, it updates the Q-values to gradually improve its policy. The Q-value can be updated by the following formula:

$$Q\left(S_c, A_c\right) \leftarrow (1-\kappa) Q\left(S_c, A_c\right) + \left(R_{c+1} + \gamma \max_a Q\left(S_{c+1}, a\right)\right) \tag{1}$$

where R_{c+1} is the reward obtained by the agent taking action A_c in state S_c, $\max_a Q\left(S_{c+1}, a\right)$ is the expectation of the maximum reward that the agent can get in state S_{c+1}.

2.2. Multi-modal optimization based on niching methods

Niching methods are inspired by the natural phenomenon of species aggregation and division into subpopulations. To deal with MMOPs, many niching-based strategies have been developed and combined with population-based optimization algorithms. In Qu et al. (2012), crowding and speciation strategy which based on neighborhood have been incorporated into differential evolution, resulting in the proposal of two multi-modal optimization algorithms (i.e., NCDE and NSDE). In NCDE, neighborhood information is utilized to identify dense regions in the population, and a crossover operation is employed that selects only relatively sparse individuals, thereby enhancing the algorithm's exploration capability and convergence speed. Different from NCDE, NSDE introduces neighborhood based speciation strategies and crowding strategies to promote the maintenance of individual diversity and high-quality solutions in populations. Inspired by the above works, Liao et al. (2023) introduced historical archival assistance strategies, speciation strategies, and variable neighborhood strategies to maintain population diversity. Xiong et al. (2023) proposed an adaptive neighborhood strategy and a species formation mechanism to make the algorithm better adapt to the diversity and complexity of optimization problems. Wang et al. (2022) developed a distributed differential evolution algorithm by employing a master slave multi-niche distributed model. In Qu et al. (2013), a distance strategy-based niching method has been introduced to cooperated with particle swarm optimizer (i.e., LIPS) to solve MMOPs. LIPS employs multiple local best particles instead of relying solely on a global best particle, thereby improving the algorithm's ability to exploit local areas and enhancing its overall search effectiveness. Inspired by this work, Wang et al. (2023) adopts an adaptive local search method to reduce the adjustment of algorithm parameters. Luo et al. (2022) proposed a nearest-better-neighbor clustering method to enhance diversity during population evolution.

3. A reinforcement learning-based neighborhood search operator and its application in population-based algorithms

In this section, the framework of the proposed RLNS is first illustrated, then the proposed reinforcement learning-based neighborhood range selection strategy, neighborhood subpopulation generation strategy and local vector encirclement model are described in detail. Finally, the RLNS is employed to combine with SSA, PSO, EO and then SSA-RLNS, PSO-RLNS and EO-RLNS are obtained.

3.1. The framework of the RLNS

Fig. 1 illustrates the search mechanisms of the RLNS. The individuals in the population can be regarded as independent agents. The agents first determine the size of the subpopulation to be generated according to the reinforcement learning-based neighborhood range selection strategy, and then generate the subpopulation according to the neighborhood subpopulation generation strategy. Finally, the position of the agents are updated in each subpopulation according to the local vector encirclement model, then a new generation of the population is obtained.

3.2. Reinforcement learning-based neighborhood range selection strategy

In search process, the number of individuals in the subpopulation $\boldsymbol{Sub_n}(\mathrm{i})$ $(i = 1, 2, 3, \ldots, n)$ is a crucial parameter, which defines the search range of the current individual and directly affects the optimization ability of the algorithm. To solve the issue which is difficult to adjust the parameter $\boldsymbol{Sub_n}$ for different optimization problems, a reinforcement learning-based neighborhood range selection strategy is proposed. In this strategy, the Q-learning algorithm is employed to adaptively adjust the size of the current individual's subpopulation,

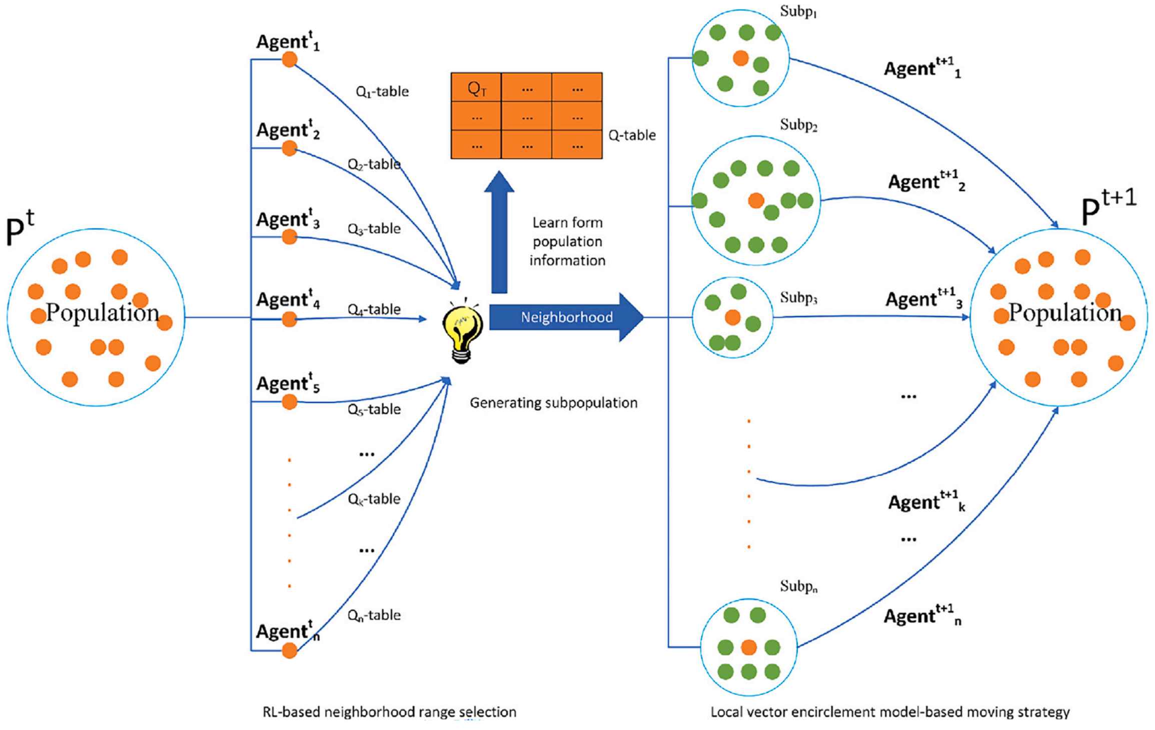


Fig. 1. Illustration of the search mechanisms of the RLNS.

Table 1
The state configuration of the environment observation.

$\boldsymbol{S}$	s_1	s_2	s_3	s_4
$\boldsymbol{e}$	e ≤ 5	5 < e ≤ 10	10 < e ≤15	e ≥ 15

Table 2
The action settings of the Q-learning.

$\boldsymbol{A}$	a_1	a_2	a_3	a_4	a_5
$\boldsymbol{Sub_n}$	5	10	20	30	40

which could effectively improve the search ability of the RLNS. The reinforcement learning-based neighborhood range selection strategy is described in detail as follows.

Suppose the population consists of n individuals, each of whom can be considered as an independent agent. The state observed by an agent is defined as the number of individuals e within a sphere centered at the agent's position with a radius of ε, where ε is a small constant. The details of state configuration are presented in Table 1, where **S** is the observed state. In general, a larger value of e indicates a greater number of individuals in the vicinity of the agent. Due to their proximity, these individuals may provide less useful position information to the agent. Consequently, the size of the subpopulation generated by the current agent is often enlarged by Q-learning under these conditions.

The search area of an agent in the RLNS is determined by the size of the subpopulation it generates. Specifically, larger subpopulation lead the agent to explore a wider range, while smaller subpopulation constrain the agent to a narrower exploration range. The Q-learning is employed to adjust the size of subpopulation generated by individuals in response to their observed states, its action settings is shown in Table 2, where **A** is the action space of the agent.

The Q-learning relies on reward as a critical input that drives updates to the Q table during the learning process and influences the optimal strategy that the algorithm learns. In the RLNS, the goal is to improve the fitness value of the updated agent compared to the current agent. Therefore, a positive reward is given when the updated agent's fitness value is better than that of the current agent, while a negative reward is given when the updated agent's fitness value is worse. The detailed description of rewards is as follows:

$$\text{Reward } = \begin{cases} 1.0 & \text{if } f^{w+1} > f^{w} \\ 0 & \text{if } f^{w+1} = f^{w} \\ -1.0 & \text{if } f^{w+1} < f^{w} \end{cases} \tag{2}$$

where f^w is the fitness value of the current agent. After the operator completes the evolution of generation w, the **Q** table is updated according to formula (1)

3.3. Neighborhood subpopulation generation strategy

In swarm intelligence algorithm, each individual in the population should have the ability to think independently, who could evaluate the similarity of other individuals to them according to specific rules. Individuals with higher similarity are more likely to be considered as members of the subpopulation by the current individual. Euclidean distance is employed to evaluate the similarity between the current individual and other individuals. The Euclidean distance between the current individual i and other individuals j $(j = 1, 2, 3, \dots, n)$ is calculated as follows:

$$\boldsymbol{Dist}_{i,j}^{w+1} = \left\| \boldsymbol{V}_i^w - \boldsymbol{V}_j^w \right\|, \tag{3}$$

where $\boldsymbol{Dist}_{i,j}^{w+1}$ $(j = 1, 2, 3, \dots, n)$ is the Euclidean distance between the ith individual and the jth individual in the current iteration, $\boldsymbol{V}_i^w$ and $\boldsymbol{V}_j^w$ are the position vectors of the ith and jth individuals, respectively. Based on the above rules, the pseudo code of the neighborhood subpopulation generation strategy is shown in Algorithm 1. In Algorithm 1, $\boldsymbol{V}$ is the position matrix of the current population, with the size of rows representing the number of individuals and the size of columns representing the dimension of the problem to be optimized. $\boldsymbol{B_k}$ $(k = 1, 2, 3, \dots, n)$ is the position matrix of the subpopulation generated by the kth individual. Taking individual i as an example, in order to generate a subpopulation belonging to individual i, the distance between individual i and other individuals in the population is first calculated (lines 2–5), and then $\boldsymbol{Sub_n}(i)$ individuals closer to individual i are selected to form a subpopulation together with individual i (lines 6–8).

Algorithm 1 Neighborhood subpopulation generation strategy

```
Require: The number of the individuals in the neighborhood subpopulation $\boldsymbol{Sub_n}(i)(i = 1,2,3,\cdots,n)$, the number of individuals in population $n$, the position of population $\boldsymbol{V}$.
Ensure: The neighborhood subpopulation $[\boldsymbol{B_1}, \boldsymbol{B_2}, \cdots, \boldsymbol{B_n}]$.
 1: for $i = 1 : n$
 2:     for $j = 1 : n$
 3:         $\boldsymbol{Dist}(i,j) \leftarrow$ Calculate Euclidean distance
 4:         between individual $i$ and individual $j$ by (3);
 5:     end for
 6:     $\boldsymbol{S} \leftarrow \boldsymbol{V}$;
 7:     Sort the $\boldsymbol{S}$ in ascending order according to $\boldsymbol{Dist}(i,:)$;
 8:     $\boldsymbol{B_i} \leftarrow \boldsymbol{S}(1 : \boldsymbol{Sub_n}(i), :)$.
 9: end for
10: return $[\boldsymbol{B_1}, \boldsymbol{B_2}, \cdots, \boldsymbol{B_n}]$.
```

3.4. A local vector encirclement model and the proposed SSA-RLNS, PSO-RLNS, EO-RLNS

For the local vector encirclement model, as shown in Fig. 3, the points represent the position of the individuals, $\boldsymbol{V_c^w}$ is the position of the agent to be updated. The individuals in the circle are the individuals in the subpopulation of $\boldsymbol{V_c^w}$. $\boldsymbol{V_{better}^w}$ and $\boldsymbol{V_{worse}^w}$ are the positions of individuals with better and worse fitness values in the subpopulation of the $\boldsymbol{V_c^w}$, respectively. It can be seen that the position of the updated agent will fall inside the parallelogram defined by vectors $\boldsymbol{V_{better}^w} - \boldsymbol{V_c^w}$ and $\boldsymbol{V_c^w} - \boldsymbol{V_{worse}^w}$. The local vector encirclement model is given as follows:

$$\boldsymbol{V_c^{w+1}} = \boldsymbol{V_c^w} + rn_6\left(\boldsymbol{V_c^w} - \boldsymbol{V_{worse}^w}\right) + rn_7\left(\boldsymbol{V_{better}^w} - \boldsymbol{V_c^w}\right), \tag{4}$$

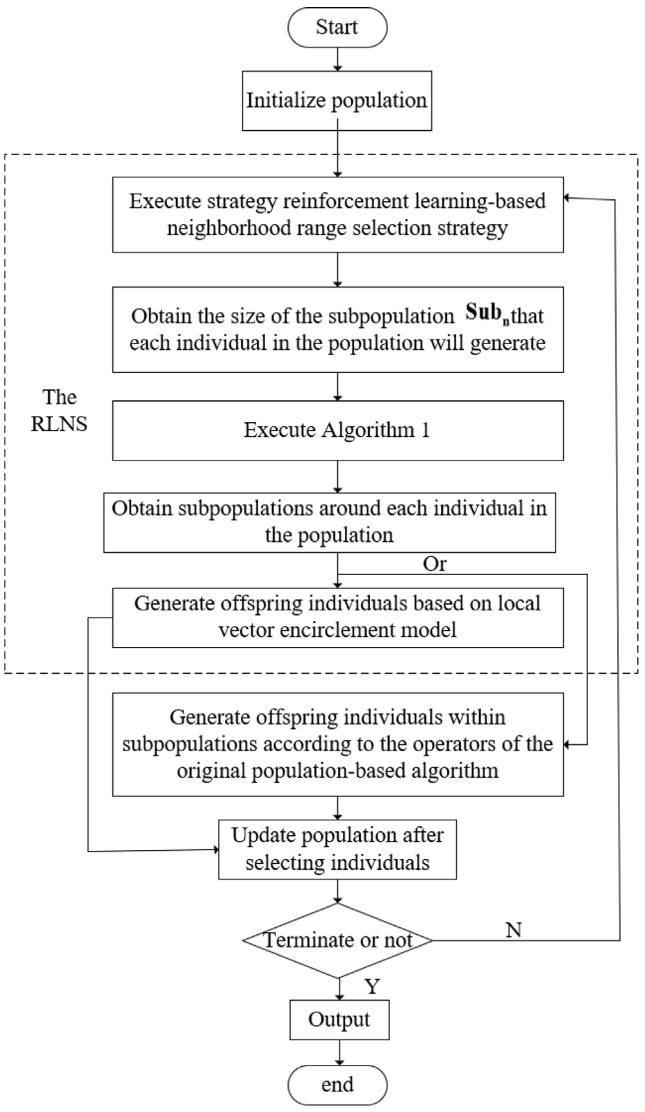
where rn_6, rn_7 $\in$ $[0,1]$ are randomly generated numbers uniformly distributed to modify the length of the vectors, *better* and *worse* are calculated by the following formula:

$$worse = \begin{cases} worst & \text{if } \left(R_b \le T_{q2}\right) \\ randW & \text{otherwise}, \end{cases}$$
$$better = \begin{cases} best & \text{if } \left(R_c \le T_{q2}\right) \\ randB & \text{otherwise}. \end{cases} \tag{5}$$

The meanings of the parameters in (5) are described as follows. Random numbers R_b, R_c $\in$ $[0,1]$ follow a uniform distribution. T_{q2} is the selection probability. The subpopulation $\boldsymbol{V_c^w}$ is sorted in ascending order based on the fitness values of k individuals. *worst* and *best* denote the indices of the positions within the neighborhood of $\boldsymbol{V_c^w}$ that correspond to the lowest and highest fitness values, respectively. Index $randB$ is chosen at random from the range of integers $[1, 0.05k]$, and index $randW$ is selected randomly from the range of integers $[0.95k, k]$. Both $randB$ and $randW$ are integer values.

In the local vector encirclement model, the utilization ratio of individuals in the subpopulation is improved, thus significantly enhances the population's capability for local search.

The process of combining RLSN with general population-based optimization algorithms is shown in Fig. 2. Firstly, the population is initialized, and then execute the proposed RLNS operator. In RLNS, the proposed reinforcement learning based neighborhood range selection strategy is used to control the size of subpopulations that each individual will generate, Algorithm 1 is employed to generate subpopulations around each individual in the population. Afterwards, the proposed local vector encirclement model or operators of general population based optimization algorithms are employed to generate offspring individuals



Start
Initialize population
The RLNS
Execute strategy reinforcement learning-based neighborhood range selection strategy
Obtain the size of the subpopulation Sub$_n$ that each individual in the population will generate
Execute Algorithm 1
Obtain subpopulations around each individual in the population
Or
Generate offspring individuals based on local vector encirclement model
Generate offspring individuals within subpopulations according to the operators of the original population-based algorithm
Update population after selecting individuals
Terminate or not
N
Y
Output
end

Fig. 2. The process of combining RLSN with general population-based optimization algorithms.

within the subpopulation. Finally, update the position of the entire population.

The SSA-RLNS, PSO-RLNS and EO-RLNS can be obtained by combining the existing SSA, PSO and EO with the proposed RLNS. The pseudo code of the SSA-RLNS, PSO-RLNS and EO-RLNS are shown in Algorithms 2, 3 and 4.

In Algorithms 2, firstly, initialize the population and $\boldsymbol{Q}$ tables, and then enter the iterative process of the SSA-RLNS (lines 1–3). The positions of some individuals named producer are updated according to Hong et al. (2022) (lines 4–6). The reinforcement learning-based neighborhood range selection strategy is employed to determine the size of subpopulations $\boldsymbol{Sub_n}(i)(i = 1,2,3,\ldots,n)$ that all individuals will generate (line 8). Subpopulations $[\boldsymbol{B_1}, \boldsymbol{B_2}, \ldots, \boldsymbol{B_n}]$ are generated by Algorithm 1 (line 9). The positions of some individuals named scrounger are updated by local vector encirclement model (lines 10–12). Finally, the positions of other individuals in the population (Hong et al., 2022) and the $\boldsymbol{Q}$ table are updated and enter the next iteration of the SSA-RLNS (lines 13–19).

In Algorithms 3, firstly, the velocity and position of the PSO, as well as the $\boldsymbol{Q}$ tables, are initialized, and then enter the iterative process of the PSO-RLNS (lines 1–3). The reinforcement learning-based

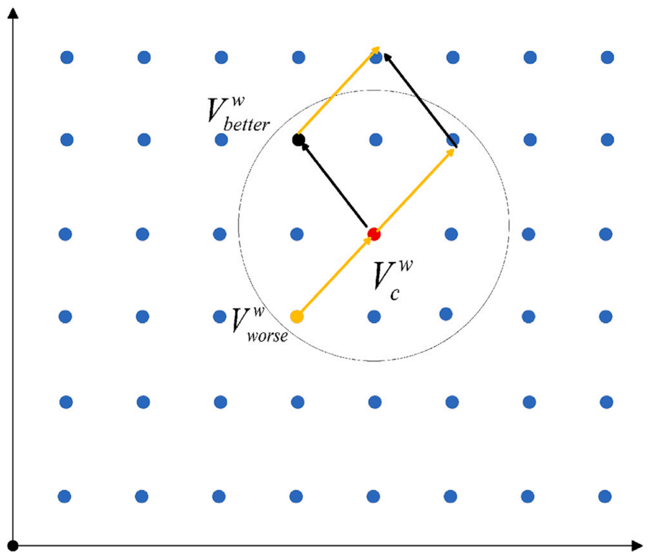


Fig. 3. The agent moves within its neighborhood.

neighborhood range selection strategy is employed to determine the size of subpopulations $\boldsymbol{Sub_n}(i)(i = 1, 2, 3, \dots, n)$ that all individuals will generate (line 4). Subpopulations $[\boldsymbol{B_1}, \boldsymbol{B_2}, \dots, \boldsymbol{B_n}]$ are generated by Algorithm 1 (line 5). The positions of individuals in the subpopulation are jointly updated by the standard PSO (Eberhart & Kenned, 1995) (lines 7–11) and the local vector encirclement model (lines 12–16). Finally, update the $\boldsymbol{Q}$ tables, select outstanding individuals, and the algorithm enters the next iteration.

In Algorithms 4, firstly, the population and the $\boldsymbol{Q}$ tables are initialized, and then enter the iterative process of the EO-RLNS (line 1). The reinforcement learning-based neighborhood range selection strategy is employed to determine the size of subpopulations $\boldsymbol{Sub_n}(i)(i = 1, 2, 3, \dots, n)$ that all individuals will generate (line 4). Subpopulations $[\boldsymbol{B_1}, \boldsymbol{B_2}, \dots, \boldsymbol{B_n}]$ are generated by Algorithm 1 (line 5). The positions of individuals in the subpopulation are jointly updated by the standard EO (Faramarzi et al., 2020) (lines 6–15) and the local vector encirclement model (lines 16–17). Finally, update the $\boldsymbol{Q}$ tables, select outstanding individuals, and the algorithm enters the next iteration.

4. Experimental results and analysis

In this section, the performance of the proposed SSA-RLNS, PSO-RLNS and EO-RLNS are tested on CEC2015 multi-niche benchmark functions which have multiple global and local optima to verify the effectiveness of the proposed RLNS in SSA, PSO and EO (Qu et al., 2014). The employed algorithms are required to find as many global optimal solutions with high precision as possible for the multi-niche benchmark functions. The parameters of the benchmark functions are shown in Table 3. At the same time, four state-of-the-art algorithms, CDE (Qu et al., 2012), CNMM (Zou et al., 2020), CLPSO (Liang et al., 2006) and WSA-IC (Zeng et al., 2020) are employed as comparison algorithms. CDE is a multimodal optimization algorithm proposed by Qu et al. which combines a neighborhood mutation strategy with the niching differential evolution algorithm. It has been demonstrated to exhibit excellent performance (Qu et al., 2012). CLPSO is a variant of PSO that utilizes the best historical information from other particles to update particle velocity. It is specifically designed for solving MMOPs and has garnered widespread attention since its introduction (Liang et al., 2006). CNMM and WSA-IC are outstanding multimodal optimization algorithms that have been proposed within the past three years. Their superiority has been confirmed in Zeng et al. (2020), Zou et al. (2020), respectively. In addition, the code for WSA-IC and

Algorithm 2 SSA-RLNS

```
Require: Maximum iterations $W$, the number of the individuals $n$, the
    safety threshold $T_{q1}$, the selection probabilities $T_{q2}$ and $T_{q3}$, the
    number of producer $P_n$, the number of vigilant $V_n$.
Ensure: Fitness value $\boldsymbol{Y_V}$, the position of population $\boldsymbol{V}$.
 1: Initialize sparrow population and $\boldsymbol{Q}$ tables, set the number of the
    iteration $W$.
 2: while $(w < W)$ do
 3: Calculating the fitness value and sort the population according to
    fitness value;
 4: for $i = 1 : P_n$
 5:     Update the producer's position according to Hong et al. (2022);
 6: end for
 7: Calculating the fitness value and sort the population according to
    fitness value;
 8: Using $Q$-Learning to calculate the size of the subpopulation of
    agents $\boldsymbol{Sub_n}(i)(i = 1, 2, 3, \cdots, n)$;
 9: Generating subpopulations $[\boldsymbol{B_1}, \boldsymbol{B_2}, \cdots, \boldsymbol{B_n}]$ by Algorithm 1;
10: for $i = P_n + 1 : n$
11:     Update the scrounger's position by (4) and (5);
12: end for
13: for $i = 1 : V_n$
14:     Update the vigilante's position according to Hong et al. (2022);
15: end for
16: Updating $\boldsymbol{Q}$ tables according to (1);
17: The position of all sparrows in the population has been updated;
18: If the updated position is better, keep it, otherwise, keep the
    original position;
19: $w = w + 1$;
20: end while
21: return $\boldsymbol{Y_V}$ and $\boldsymbol{V}$.
22: The detailed rule of the SSA is shown (Hong et al., 2022).
```

CLPSO in this paper is sourced from the author's source code, and the experimental parameters for CDE and CNMM are taken from the author's recommended parameters. The parameters of the comparison algorithms are shown in Table 4, and more details can be seen in Liang et al. (2006), Qu et al. (2012), Zeng et al. (2020), Zou et al. (2020). Note that the employed algorithms and the SSA-RLNS, PSO-RLNS, EO-RLNS runs independently on each problem for 30 times, with the number of population of 1000 and the maximum number of iterations of 500.

Statistics of all experimental results are recorded in Tables 5–6. ANFO is the mean value of the number of global peaks found by the above algorithms. Success rate shows the percentage of times that all global optima found in the total number of runs and peak ratio illustrates the average percentage of all known global optima found in the total number of runs (Qu et al., 2014). It can be seen that all the global optima of F1, F2, F5, F6 and F7 can be found by SSA-RLNS in the 30 times experiments. In addition, SSA-RLNS also performs well on other test problems. All the global optima of F2, F4, F7 and F9 can be found by PSO-RLNS. EO-RLNS performs well on F2, F3, F4, F5, F7, F8, F9 and F10. To fully compare the performance of SSA-RLNS, PSO-RLNS, EO-RLNS, CDE, CLPSO, CNMM and WSA-IC on CEC2015 multi-niche benchmark functions, the box-plot of the experimental results are shown in Figs. 5–6. It is seen that in most test problems, the data points of SSA-RLNS are significantly higher and more convergent in the box plot compared to other algorithms, which illustrates that SSA-RLNS could find more global optima and has strong robustness. To investigate the convergence speed of the algorithm, the test results of SSA-RLNS, PSO-RLNS, CDE, CNMM, and WSA-IC on F1, F3, F8, and F9 were selected to draw the convergence curves, as shown in Fig. 4. The convergence curves of EO-RLSN are not plotted in the graph because its fitness values varied widely, making it difficult to observe

Table 3
Test functions.

No.	Function name	Dimension	Peaks global/local	fmin
F1	CEC2015 MultiNich-F1	5	1/15	100
F2	CEC2015 MultiNich-F2	2	4/21	200
F3	CEC2015 MultiNich-F2	5	32/0	200
F4	CEC2015 MultiNich-F3	2	25/0	300
F5	CEC2015 MultiNich-F4	5	1/15	400
F6	CEC2015 MultiNich-F4	10	1/55	400
F7	CEC2015 MultiNich-F5	2	25/0	500
F8	CEC2015 MultiNich-F6	4	16/0	600
F9	CEC2015 MultiNich-F7	6	8/0	700
F10	CEC2015 MultiNich-F8	2	36/0	800
Search Range:[−100,100]				
level of accuracy = 0.1				

Algorithm 3 PSO-RLNS

```
Require: Maximum iterations $W$, the number of the individuals $n$, the inertia factor $\omega$, the acceleration constants $c_1$ and $c_2$.
Ensure: Fitness value $\boldsymbol{Y_V}$, the position of population $\boldsymbol{V}$.
1: Initialize the position, velocity and $\boldsymbol{Q}$ tables, set the maximum number of the iteration $W$.
2: while $(w < W)$ do
3: Calculating the fitness value;
4: Using $Q$-Learning to calculate the size of the subpopulation of agents $\boldsymbol{Sub_n}(i)(i = 1, 2, 3, \cdots, n)$;
5: Generating subpopulation $[\boldsymbol{B_1}, \boldsymbol{B_2}, \cdots, \boldsymbol{B_n}]$ by Algorithm 1;
6: for $i = 1 : n$
7:     if rand(1)< 0.5
8:             Update individual's position and velocity according to the rules of
9:             standard PSO which the $V_{better}$ calculated by (4) and (5) is instead
10:            of $gbest$ in standard PSO (Eberhart & Kenned, 1995);
11:    else
12:            Update individual's position according to (4);
13:            Update individual's velocity according to the rules of
14:            standard PSO which the $\boldsymbol{V_{better}}$ calculated by (4) and (5) is instead
15:            of $\boldsymbol{gbest}$ in standard PSO (Eberhart & Kenned, 1995);
16:    end if
17:    Updating $\boldsymbol{Q}$ table of the current individual according to (1);
18: end for
19: The position and velocity of all individuals has been updated;
20: If the updated position is better, keep it, otherwise, keep the original position;
21: $w = w + 1$;
22: end while
23: return $\boldsymbol{Y_V}$ and $\boldsymbol{V}$.
24: Standard PSO is shown in Eberhart and Kenned (1995).
```

Algorithm 4 EO-RLNS

```
Require: Maximum iterations $W$, the number of the individuals $n$, set parameters $GP = 0.5, a_1 = 2, a_2 = 1$.
Ensure: Fitness value $\boldsymbol{Y_V}$, the position of population $\boldsymbol{V}$.
1: Initialize the population and $\boldsymbol{Q}$ tables, set the number of the iteration $W$.
2: while $(w < W)$ do
3: Calculating the fitness value and arrange the population $\boldsymbol{V}$ in descending order of fitness value;
4: Using $Q$-Learning to calculate the size of the subpopulation of agents $\boldsymbol{Sub_n}(i)(i = 1, 2, 3, \cdots, n)$;
5: Generating subpopulations $[\boldsymbol{B_1}, \boldsymbol{B_2}, \cdots, \boldsymbol{B_n}]$ by Algorithm 1;
6: for $i = 1 : n$
7:     arrange the population $\boldsymbol{B_i}$ in descending order of fitness value;
8:     $\vec{C}_{eq1} = \boldsymbol{B_i}(1, :), \vec{C}_{eq2} = \boldsymbol{B_i}(2, :), \vec{C}_{eq3} = \boldsymbol{B_i}(3, :), \vec{C}_{eq4} = \boldsymbol{B_i}(4, :)$;
9:     $\vec{C}_{ave} = (\vec{C}_{eq1} + \vec{C}_{eq2} + \vec{C}_{eq3} + \vec{C}_{eq4})/4$;
10:    the $\vec{C}_{eq,pool} = [\vec{C}_{eq1}, \vec{C}_{eq2}, \vec{C}_{eq3}, \vec{C}_{eq4}, \vec{C}_{ave}]$ (Faramarzi et al., 2020);
11:    if rand(1)< 0.5
12:        Update individual's position according to the
13:        rules for the position of standard equilibrium
14:        optimizer (Faramarzi et al., 2020);
15:    else
16:        Update individual's position according to (4);
17:    end if
18:    Updating $\boldsymbol{Q}$ table of the current individual according to (1);
19: end for
20: The position of all individuals in the population has been updated;
21: If the updated position is better, keep it, otherwise, keep the original position;
22: $w = w + 1$;
23: end while
24: return $\boldsymbol{Y_V}$ and $\boldsymbol{V}$.
25: The meaning of symbols is shown in Faramarzi et al. (2020).
```

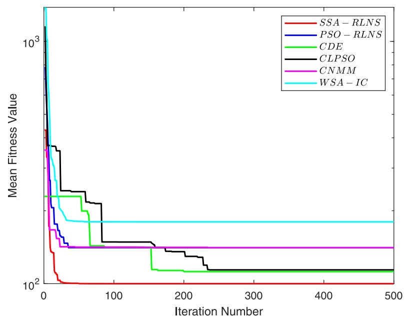
Table 4
Parameters setting of algorithms.

Algorithms	Parameters
CDE	CR = 0.9, F = 0.9
CNMM	CR = 0.9, F = 0.5, e = 0.5, r = 0.1
CLPSO	C_1 = 1.49445, C_2 = 1.49445
WSA-IC	ρ_0 = 2, η = 0, T_s = 100*n, T_f = k
SSA-RLNS	p = 0.2, k = 20
PSO-RLNS	C_1 = 1.49445, C_2 = 1.49445, ω = 1
EO-RLNS	a_1 = 5, a_2 = 1, GP = 0.5

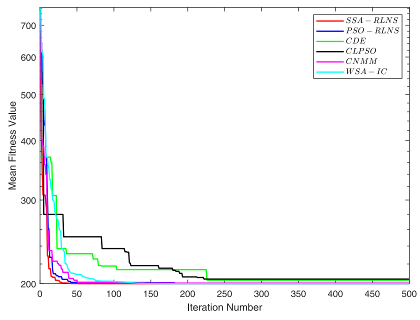
the convergence curves of other algorithms. It can be seen that SSA-RLNS and PSO-RLSN improved based on RLNS have higher convergence rates.

In RLNS, T_{q2} in Eq. (5) is the only parameter required to be tuned that determines the search direction of an individual within its subpopulation, and it consequently has a significant impact on the algorithm's performance. To analyze the impact of parameter T_{q2} on algorithm performance, we take EO-RLNS as an example. The parameters of the EO algorithm itself are sourced from Faramarzi et al. (2020), and we vary T_{q2} as 0.2, 0.5, and 0.8 to test the performance of EO-RLNS on the CEC2015 multi-niche benchmark functions. The experimental results are shown in the Table 9. It can be seen that the overall performance of the algorithm fluctuates when T takes different parameters. Specifically, when T_{q2} takes 0.8, the algorithm exhibits the best and most stable performance.

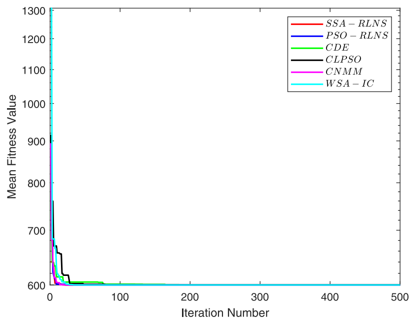
In RLNS, the number of individual in the subpopulation Sub_n is a critical parameter. In order to analyze the influence of the setting mode



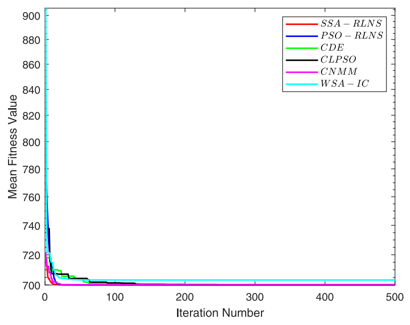
(a) F1



(b) F3



(c) F8



(d) F9

Fig. 4. Convergence curves of algorithms under comparison on the F1, F3, F8, F9.

Table 5
The performance of algorithms for F1–F5.

Functions	Criteria/algorithm	SSA-RLNS	PSO-RLNS	EO-RLNS	CDE	CLPSO	CNMM	WSA-IC
F1	SuccessRate	**1.0000**	0.0000	0.9000	0.0000	0.0000	0.0000	0.0000
	PeakRatio	**1.0000**	0.0000	0.9000	0.0000	0.0000	0.0000	0.0000
	ANOF	**1.0000**	0.0000	0.9000	0.0000	0.0000	0.0000	0.0000
	Mean run time	42.3529	120.8781	33.5124	298.0867	5.3508	322.9600	6.1620
F2	SuccessRate	**1.0000**	1.0000	1.0000	0.8333	1.0000	0.7000	0.0000
	PeakRatio	**1.0000**	1.0000	1.0000	0.9417	1.0000	0.8833	0.6583
	ANOF	**4.0000**	4.0000	4.0000	3.7667	4.0000	3.5333	2.6330
	Mean run time	4.5650	1.1802	18.3831	59.9345	4.9920	6.0825	5.1324
F3	SuccessRate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PeakRatio	0.5800	0.0000	**0.5844**	0.0000	0.0000	0.0000	0.0375
	ANOF	18.5600	0.0000	**18.7000**	0.0000	0.0000	0.0000	1.4000
	Mean run time	23.4989	119.8438	34.5456	306.8072	5.1792	312.5012	6.0924
F4	SuccessRate	**1.0000**	1.0000	0.0000	1.0000	1.0000	0.7330	0.0000
	PeakRatio	**1.0000**	1.0000	0.9600	1.0000	1.0000	0.7542	0.2562
	ANOF	**25.0000**	25.0000	24.0000	25.0000	25.0000	24.0000	8.2000
	Mean run time	3.5521	0.7768	19.5486	11.8767	4.9452	6.1012	6.1620
F5	SuccessRate	**1.0000**	0.0000	0.9000	0.0000	0.1111	0.0000	0.0000
	PeakRatio	**1.0000**	0.0000	0.9000	0.0000	0.1000	0.0000	0.0000
	ANOF	**1.0000**	0.0000	0.9000	0.0000	0.1000	0.0000	0.0000
	Mean run time	32.9948	122.2813	72.8385	307.6496	5.2167	322.9600	6.1870

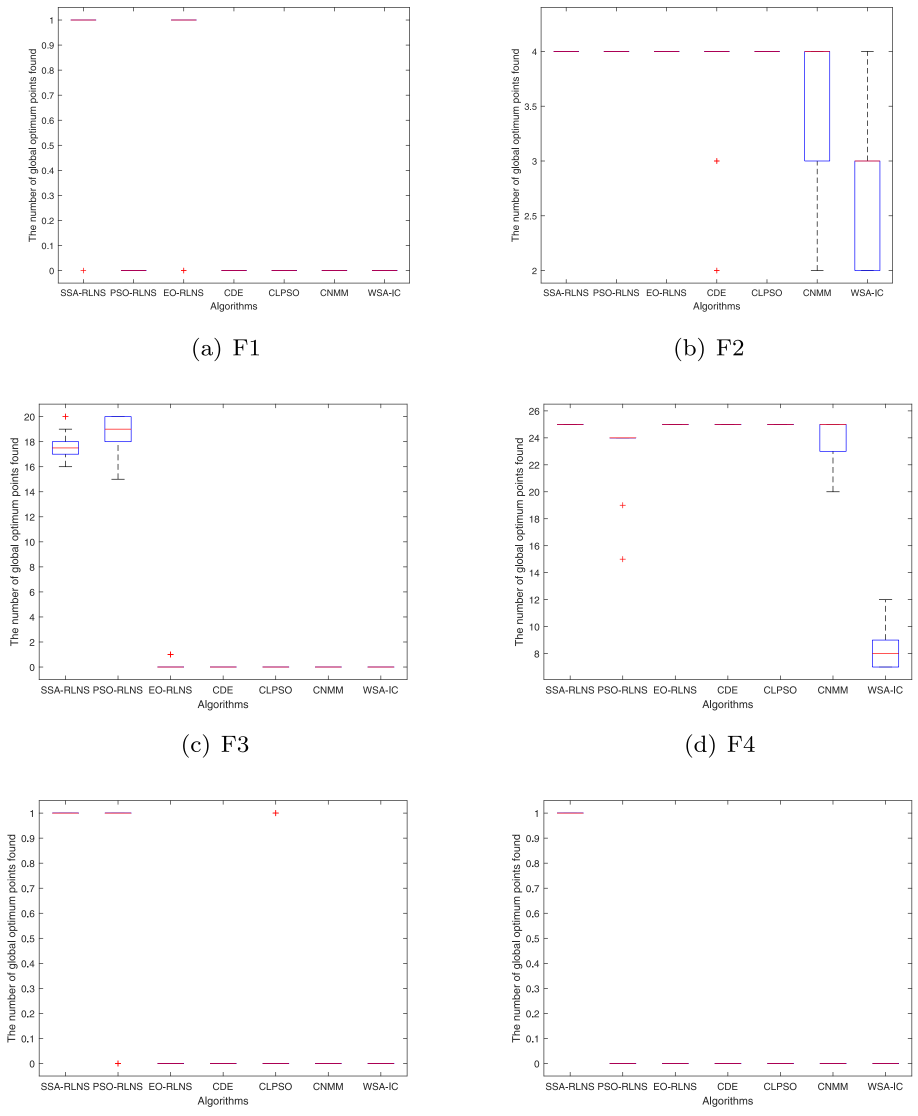
of parameter Sub_n on the algorithm effect, the performance of SSA-RLNS with fixed parameter Sub_n of 5, 10, 20, 30 and reinforcement learning-based neighborhood range selection strategy are tested and the experimental results are recorded in Table 7–8. It can be seen that the SSA-RLNS with reinforcement learning-based neighborhood range selection strategy performs best on the CEC2015 multi-niche benchmark functions. Furthermore, Fig. 7 illustrates the search process of SSA-RLNS on F3. There are 1000 individuals who are asked to locate the global optima. During iteration, each individual will search in its own neighborhood. After 80 iterations, all the global optima have been found successfully.

5. Application of the SSA-RLNS in inverse kinematics of robot manipulator

In this section, the proposed SSA-RLNS is employed to solve the IK of the RM. Firstly, the kinematics description of the RM is briefly introduced, and then the method of solving the IK of the RM by the multi-modal optimization algorithm is described in detail. Finally,

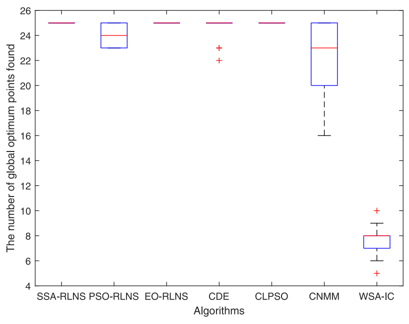
Table 6
The performance of algorithms for F6–F10.

Functions	Criteria/algorithm	SSA-RLNS	PSO-RLNS	EO-RLNS	CDE	CLPSO	CNMM	WSA-IC
F6	SuccessRate	**1.0000**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PeakRatio	**1.0000**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	ANOF	**1.0000**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	Mean run time	265.8881	139.0469	158.2031	310.5412	5.5068	305.0600	6.1178
F7	SuccessRate	**1.0000**	1.0000	0.3000	0.9000	1.0000	0.4333	0.0000
	PeakRatio	**1.0000**	1.0000	0.9560	0.9907	1.0000	0.8880	0.3013
	ANOF	**25.0000**	25.0000	23.9000	24.7667	25.0000	22.2000	7.5333
	Mean run time	1.0373	1.2548	13.1434	4.7513	4.9764	3.0487	4.9541
F8	SuccessRate	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PeakRatio	**0.9600**	0.5125	0.8438	0.0771	0.5292	0.0000	0.1083
	ANOF	**15.3600**	8.2000	13.5000	1.2333	8.4700	0.0000	1.7333
	Mean run time	8.9203	5.2911	6.8848	271.6071	4.9967	311.2844	5.6093
F9	SuccessRate	0.3000	**1.0000**	0.2000	0.0000	1.0000	0.0000	0.0000
	PeakRatio	0.9000	**1.0000**	0.8375	0.2583	1.0000	0.6094	0.1333
	ANOF	7.200	**8.0000**	6.7000	2.0667	8.0000	4.1250	1.0067
	Mean run time	15.4001	13.2188	8.4445	245.2492	5.2260	68.1117	5.9202
F10	SuccessRate	0.0000	0.0000	0.0000	**1.0000**	0.0000	0.0000	0.0000
	PeakRatio	0.7368	0.8611	0.6833	**1.0000**	0.6093	0.6667	0.1685
	ANOF	26.1000	31.0000	24.6000	**36.0000**	21.9333	24.0000	6.0667
	Mean run time	42.3529	0.2339	2.5509	5.9776	4.8407	322.9600	5.1496

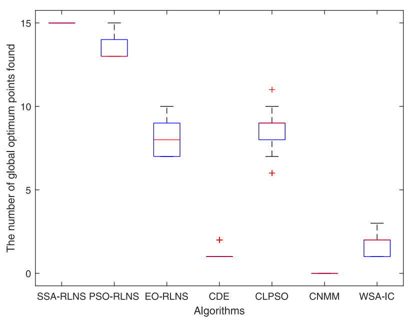


(a) F1
(b) F2
(c) F3
(d) F4
(e) F5
(f) F6

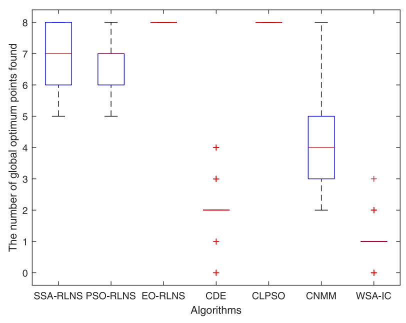
Fig. 5. Box plots of experimental results on F1–F6.



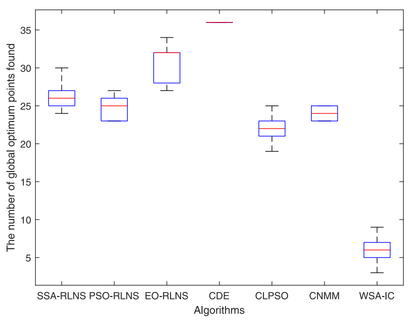
(a) F7



(b) F8



(c) F9



(d) F10

Fig. 6. Box plots of experimental results on F7–F10.

Table 7
The performance of SSA-RLNS for F1–F5.

Functions	Criteria/*Sub*	RL	*Sub* = 5	*Sub* = 10	*Sub* = 20	*Sub* = 30
F1	SuccessRate	1.00	1.00	1.00	1.00	1.00
	PeakRatio	1.00	1.00	1.00	1.00	1.00
	ANOF	1.00	1.00	1.00	1.00	1.00
F2	SuccessRate	1.00	1.00	1.00	1.00	1.00
	PeakRatio	1.00	1.00	1.00	1.00	1.00
	ANOF	4.00	4.00	4.00	4.00	4.00
F3	SuccessRate	0.00	0.00	0.00	0.00	0.00
	PeakRatio	0.58	0.03	0.14	0.40	0.31
	ANOF	17.80	1.00	4.50	12.90	10.10
F4	SuccessRate	1.00	0.80	0.70	0.00	0.00
	PeakRatio	1.00	0.99	0.99	0.63	0.72
	ANOF	25.00	24.80	24.70	15.90	17.90
F5	SuccessRate	1.00	0.40	0.80	0.90	0.80
	PeakRatio	1.00	0.40	0.80	0.90	0.80
	ANOF	1.00	0.40	0.80	0.90	0.80

Table 8
The performance of SSA-RLNS for F6–F10.

Functions	Criteria/*Sub*	RL	*Sub* = 5	*Sub* = 10	*Sub* = 20	*Sub* = 30
F6	SuccessRate	1.00	0.30	0.00	0.10	0.00
	PeakRatio	1.00	0.30	0.00	0.10	0.00
	ANOF	1.00	0.30	0.00	0.10	0.00
F7	SuccessRate	1.00	0.90	0.30	0.00	0.00
	PeakRatio	1.00	0.99	0.96	0.64	0.77
	ANOF	25.00	24.77	24.00	16.00	19.20
F8	SuccessRate	0.00	0.00	0.60	0.00	0.00
	PeakRatio	0.96	0.50	0.96	0.82	0.69
	ANOF	15.20	8.00	15.60	13.10	11.00
F9	SuccessRate	0.40	0.00	0.60	0.20	0.00
	PeakRatio	0.88	0.33	0.95	0.86	0.60
	ANOF	7.20	2.60	7.60	6.90	4.80
F10	SuccessRate	0.00	1.00	0.00	0.00	0.00
	PeakRatio	0.84	0.71	0.67	0.53	0.52
	ANOF	26.20	25.60	24.20	18.90	18.80

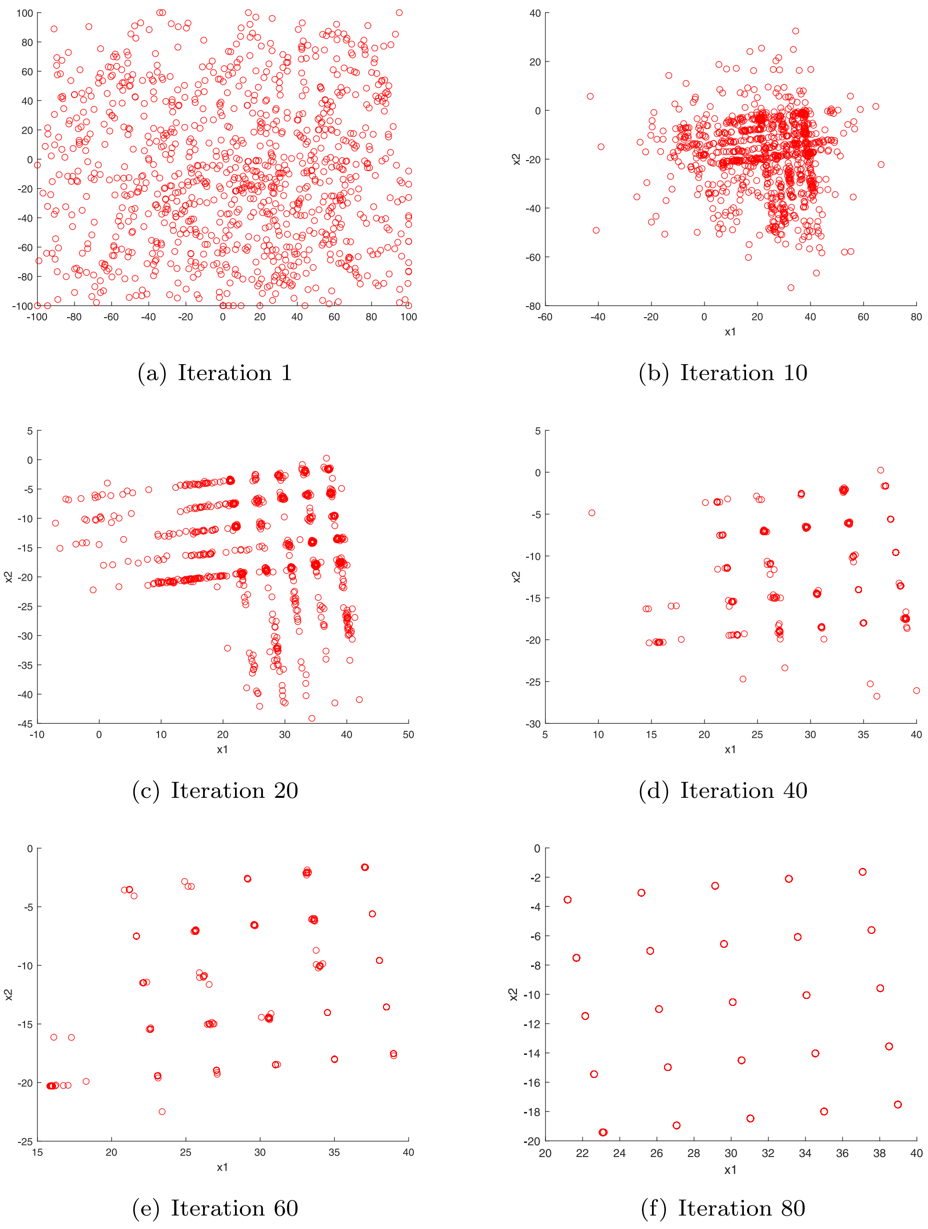


Fig. 7. Position of sparrow of SSA-RLNS in various iterations.

Table 9
Experimental results of EO-RLNS at T values of 0.2, 0.5, and 0.8, respectively.

Functions	Criteria/T_{q2}	0.2	0.5	0.8	Functions	Criteria/T_{q2}	0.2	0.5	0.8
F1	SuccessRate	0.7000	0.9000	0.9000	F6	SuccessRate	0.0000	0.0000	0.0000
	PeakRatio	0.7000	0.9000	0.9000		PeakRatio	0.0000	0.0000	0.0000
	ANOF	0.7000	0.9000	0.9000		ANOF	0.0000	0.0000	0.0000
F2	SuccessRate	1.0000	1.0000	1.0000	F7	SuccessRate	0.2000	0.3000	1.0000
	PeakRatio	1.0000	1.0000	1.0000		PeakRatio	0.9640	0.9560	1.0000
	ANOF	4.0000	4.0000	4.0000		ANOF	24.1000	23.9000	25.0000
F3	SuccessRate	0.0000	0.0000	0.0000	F8	SuccessRate	0.2000	0.0000	0.0000
	PeakRatio	0.6250	0.5844	0.7440		PeakRatio	0.9000	0.8438	0.8563
	ANOF	20.0000	18.7000	18.6000		ANOF	14.4000	13.5000	13.7000
F4	SuccessRate	0.3000	0.0000	0.2000	F9	SuccessRate	0.1000	0.2000	0.1000
	PeakRatio	0.9600	0.9600	0.9440		PeakRatio	0.8000	0.8375	0.8500
	ANOF	24.0000	24.0000	23.6000		ANOF	6.4000	6.7000	6.8000
F5	SuccessRate	1.0000	0.9000	0.9000	F10	SuccessRate	0.0000	0.0000	0.0000
	PeakRatio	1.0000	0.9000	0.9000		PeakRatio	0.6833	0.6833	0.6900
	ANOF	1.0000	0.9000	0.9000		ANOF	24.6000	24.6000	24.8000

SSA-RLNS and several excellent multi-modal optimization algorithms are used to solve the IK of RM, which verifies the effectives of the SSA-RLNS.

5.1. Kinematic descriptions of robot manipulator

A series of rigid bodies are connected by joints to form a RM (Kucuk & Bingul, 2014; Rokbani et al., 2022). The computation of the posture of the end effector given the angle of each joint variable is called forward kinematics. The kinematics chain of a n-DOF RM is shown in Fig. 8, where L_t $(t = 1, 2, 3, \dots, n)$ is the tth linkage of the RM. The forward kinematics of a n-DOF RM is calculated as follows (Gao et al., 2014):

$$\begin{aligned} {}^0T_n &= {}^0T_1(\theta_1)\,{}^1T_2(\theta_2)\,{}^2T_3(\theta_3) \quad \dots \quad {}^{n-1}T_n(\theta_n) \\ &= \prod_{t=1}^{n} {}^{t-1}T_t(\theta_t), \end{aligned} \tag{6}$$

where ${}^0T_n$ is the homogeneous transformation matrix which contains the position and orientation of the end effector, n is the number of degrees of freedom of the RM, θ_t represents the rotation angle of the tth $(t = 1, 2, 3, \dots, n)$ joint. The Denavit–Hartenberg model is employed to establish the kinematic of the RM. In the Denavit–Hartenberg model, the posture of the end effector in the fixed coordinate system can be expressed by ${}^0T_n$, and the relationship of position between the tth and the $(t-1)$th linkage can be represented by ${}^{t-1}T_t$. The transformation matrix ${}^{t-1}T_t$ can be calculated by

$${}^{t-1}T_t(\theta_t) = \begin{bmatrix} c\theta_t & -s\theta_t c\alpha_t & s\theta_t s\alpha_t & a_t c\theta_t \\ s\theta_t & c\theta_t c\alpha_t & -c\theta_t s\alpha_t & a_t s\theta_t \\ 0 & s\alpha_t & c\alpha_t & d_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where d_t, α_t, θ_t and a_t are the parameters of the RM. These parameters are linkage offset, linkage twist, linkage angle and linkage length, respectively. For simplicity, the letters s and c are instead of $\sin$ and $\cos$ operations and the ${}^0T_n$ can be written as follows:

$$\begin{aligned} {}^0T_n &= \begin{bmatrix} e_x & f_x & g_x & h_x \\ e_y & f_y & g_y & h_y \\ e_z & f_z & g_z & h_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Q} & \mathbf{E} \\ \mathbf{0} & 1 \end{bmatrix}, \end{aligned} \tag{8}$$

where $\mathbf{Q}$ is the rotation matrix, representing the orientation of the end effector of RM, and $\mathbf{E}$ is the position of the RM of end effector in the fixed coordinate system.

5.2. Inverse kinematics is solved by SSA-RLNS

Suppose that an individual in the heuristic optimization algorithm represents joint variables $\mathbf{G} = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$, where θ_i $(i = 1, 2, 3, \dots, n)$ is the rotation angle of each joint of the RM, the homogeneous transformation matrix ${}^0T_n$ of the joint variables $\mathbf{G}$ can be calculated by forward kinematics directly. The desired posture of the RM's end effector can be expressed by the following matrix:

$$\begin{aligned} {}^0\hat{T}_n &= \begin{bmatrix} \hat{e}_x & \hat{f}_x & \hat{g}_x & \hat{h}_x \\ \hat{e}_y & \hat{f}_y & \hat{g}_y & \hat{h}_y \\ \hat{e}_z & \hat{f}_z & \hat{g}_z & \hat{h}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ &= \begin{bmatrix} \hat{\mathbf{Q}} & \hat{\mathbf{E}} \\ \mathbf{0} & 1 \end{bmatrix} \end{aligned} \tag{9}$$

and then, the objective function can be constructed as follows:

$$f\left({}^0\hat{T}_n, {}^0T_n\right) = \left\| {}^0\hat{T}_n - {}^0T_n \right\|, \tag{10}$$

Table 10
The DH table of puma560 of 6DOF.

Joint	θ (rad)	d (m)	a (m)	α (rad)
1	θ_1	0.0	0.0	$\pi/2$
2	θ_2	0.0	0.431	0
3	θ_3	0.15	0.0203	$-\pi/2$
4	θ_4	0.4318	0.0	$\pi/2$
5	θ_5	0.0	0.0	$-\pi/2$
6	θ_6	0.0	0.0	0.0

where $f\left({}^0\hat{T}_n, {}^0T_n\right)$ is the fitness function, which is used to measure the error between desired matrix ${}^0\hat{T}_n$ and matrix ${}^0T_n$. Furthermore, $\hat{\mathbf{Q}}$ and $\hat{\mathbf{E}}$ in $\hat{T}_n$ represent the position and orientation of the end effector of the RM respectively. The fitness function can also be written as follows:

$$f\left({}^0\hat{T}_n, {}^0T_n\right) = \alpha\|\hat{\mathbf{Q}} - \mathbf{Q}\| + (1-\alpha)\|\hat{\mathbf{E}} - \mathbf{E}\|, \tag{11}$$

where α is the weighting coefficient. Solving the IK of RM can be seen as an optimization problem that minimizes the error between the target posture and the actual posture (i.e., $\min f\left({}^0\hat{T}_n, {}^0T_n\right)$), which is referred to as a MMOP with multiple global optimal solutions. The multimodal optimization algorithm is employed to solve the IK of a RM, and its solving process can be summarized as follows.

The first step is population initialization, where each individual $\mathbf{G_i} = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$ ($\mathbf{G_i}$ represents the ith individual in the population) in the population represents a set of joint angles of the RM, and the posture ${}^0T_{in}$ of the end effector of RM corresponding to $\mathbf{G_i} = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$ can be calculated by forward kinematics.

The second step is to calculate the posture of each individual's corresponding end effector of the RM based on forward kinematics and calculate the error between it and the target posture $f\left({}^0\hat{T}_n, {}^0T_n\right)$.

In the final step, the algorithm enters the iterative optimization process and searches for the individual whose objective function value $f\left({}^0\hat{T}_n, {}^0T_n\right)$ is 0, that is, the IK solutions of the RM.

Take puma560 as an example, the new proposed SSA-RLNS is applied to the IK of the puma560. The Denavit–Hartenberg table of the puma560 is shown in Table 10. To compare the performance of multi-model optimization algorithms in IK of the puma560, the multi-modal algorithms such as CDE, CNMM, CLPSO, WSA-IC are employed as comparison algorithms.

Thirty groups of the end effector posture are generated randomly in the reachable space of puma560. The employed multi-model optimization algorithms are required to find as many inverse solutions corresponding to posture of the end effector of puma560 as possible. To ensure fairness, the number of population of all algorithms is 1000, and the maximum number of iterations is 300. The experimental results are recorded in the Table 11, the terms Ave, Best, Std, and Worst refer to the average, best, standard deviation, and worst values obtained from the statistical analysis of the aforementioned algorithms. It can be seen that SSA-RLNS could always locate multiple inverse solutions of the RM successfully.

In a word, for the same end posture of the RM, SSA-RLNS can always successfully obtain multiple inverse solutions, which is of great significance for trajectory planning, trajectory tracking and other research.

6. Conclusion

In this paper, a RLNS has been proposed for multi-modal optimization problems. In the RLNS, a reinforcement learning-based neighborhood range selection strategy, a neighborhood subpopulation generation strategy, and a local vector encirclement model have been designed to guide the movement patterns of agents. In order to validate the efficacy the proposed RLNS, the SSA-RLNS, PSO-RLNS and EO-RLNS have been proposed by combining RLNS with the existing SSA, PSO and EO. The existing state-of-the-art multi-modal optimization algorithms have been employed to compare with the proposed SSA-RLNS, PSO-RLNS and EO-RLNS to verify the performance of the RLNS in SSA, PSO

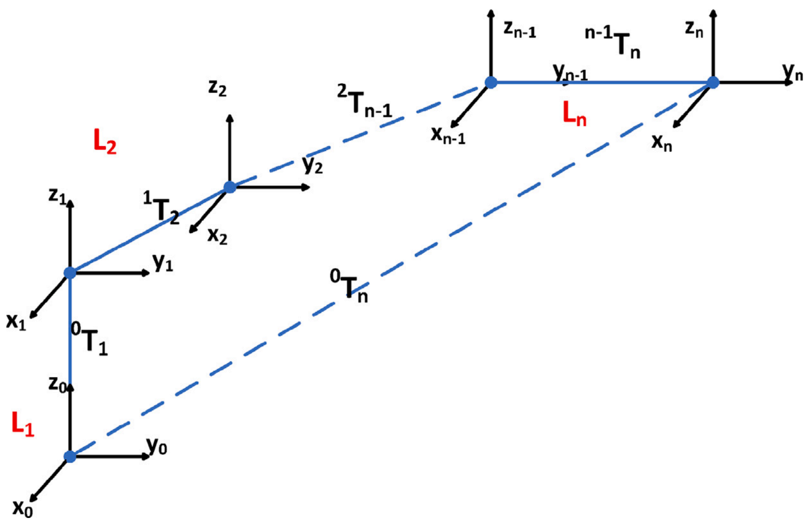


Fig. 8. The kinematic chain of a *n*-DOF robot manipulator.

Table 11
The experimental record of the inverse kinematics solution by multi-model algorithms.

Criteria\Algorithm	SSA-RLNS	CNMM	WSA-IC	CDE	CLPSO
Ave	**6.033**	0	1.3333	0	0.8667
Std	1.6291	0	0.4795	0	0.8604
Median	**6**	0	1	0	1
Worst	**3**	0	0	0	0
Best	**8**	0	2	0	3

and EO. The experimental results in CEC2015 multi-niche benchmark functions have shown that SSA-RLNS, PSO-RLNS and EO-RLNS have a significant improvement over the employed multi-modal optimization algorithms. Further, the necessity of the reinforcement learning-based neighborhood range selection strategy in RLNS has been demonstrated by parameter sensitivity analysis. Finally, the SSA-RLNS has been successfully used to solve the IK of the RM. The simulation results on puma560 have shown that the SSA-RLNS can provide multiple joint solutions leading to the same posture of the end effector with high accuracy.

CRediT authorship contribution statement

Jiale Hong: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Bo Shen:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Funding acquisition. **Anqi Pan:** Conceptualization, Methodology, Project administration, 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 was supported in part by the National Natural Science Foundation of China under Grants 62273088 and 62303108, and the Shanghai Sailing Program, China under Grant 21YF1401400.

References

Das, S., Abraham, A., Chakraborty, U. K., & Konar, A. (2009). Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, *13*(3), 526–553.

Dereli, S., & Koker, R. (2020). A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. *Artificial Intelligence Review*, *53*(2), 949–964.

Duc, L. A., Li, K., Nguyen, T. T., Yen, V. M., & Truong, T. K. (2018). A new effective operator for the hybrid algorithm for solving global optimisation problems. *International Journal of Systems Science*, *49*(5), 1088–1102.

Eberhart, R., & Kenned, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43).

El-Sherbiny, A., Elhosseini, M. A., & Haikal, A. Y. (2018). A comparative study of soft computing methods to solve inverse kinematics problem. *Ain Shams Engineering Journal*, *9*(4), 2535–2548.

Epitropakis, M. G., Tasoulis, D. K., Pavlidis, N. G., Plagianakos, V. P., & Vrahatis, M. N. (2011). Enhancing differential evolution utilizing proximity-based mutation operators. *IEEE Transactions on Evolutionary Computation*, *15*(1), 99–119.

Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-based Systems*, *191*.

Farshi, T. R., & Orujpour, M. (2021). A multi-modal bacterial foraging optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *12*(11), 10035–10049.

Gao, W., Yen, G. G., & Liu, S. (2014). A cluster-based differential evolution with self-adaptive strategy for multimodal optimization. *IEEE Transactions on Cybernetics*, *44*(8), 1314–1327.

Hernandez-Barragan, J., Lopez-Franco, C., Arana-Daniel, N., Alanis, A. Y., & Lopez-Franco, A. (2021). A modified firefly algorithm for the inverse kinematics solutions of robotic manipulators. *Integrated Computer-Aided Engineering*, *28*(3), 257–275.

Hong, J., Shen, B., Xue, J., & Pan, A. (2022). A vector-encirclement-model-based sparrow search algorithm for engineering optimization and numerical optimization problems. *Applied Soft Computing*, *131*.

Jiang, R., Zhang, J., Tang, Y., Feng, J., & Wang, C. (2022). Self-adaptive DE algorithm without niching parameters for multi-modal optimization problems. *Applied Intelligence*, *52*(11), 12888–12923.

Kucuk, S., & Bingul, Z. (2014). Inverse kinematics solutions for industrial robot manipulators with offset wrists. *Applied Mathematical Modelling*, *38*(7–8), 1983–1999.

Li, J., Balazs, M., Parks, G., & Clarkson, P. (2002). A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, *10*(3), 207–234.

Li, X., Epitropakis, M. G., Deb, K., & Engelbrecht, A. (2017). Seeking multiple solutions: an updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, *21*(4), 518–538.

Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*(3), 281–295.

Liao, Z., Mi, X., Pang, Q., & Sun, Y. (2023). History archive assisted niching differential evolution with variable neighborhood for multimodal optimization. *Swarm and Evolutionary Computation*, *76*.

Lin, C.-Y. (2022). Fuzzy AHP-based prioritization of the optimal alternative of external equity financing for start-ups of lending company in uncertain environment. *Romanian Journal of Information Science and Technology*, *25*, 133–149.

Lin, X., Luo, W., & Xu, P. (2021). Differential evolution for multimodal optimization with species by nearest-better clustering. *IEEE Transactions on Cybernetics*, *51*(2), 970–983.

Luo, J., & Gu, F. (2016). An adaptive niching-based evolutionary algorithm for optimizing multi-modal function. *International Journal of Pattern Recognition and Artificial Intelligence*, *30*(3).

Luo, W., Qiao, Y., Lin, X., Xu, P., & Preuss, M. (2022). Hybridizing niching, particle swarm optimization, and evolution strategy for multimodal optimization. *IEEE Transactions on Cybernetics*, *52*(7), 6707–6720.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.

Mohammadi, G., & Moaddabi, E. (2021). Using two metaheuristic algorithms for scheduling parallel machines with sequence dependent set-up times in job shop industries. *International Journal of Systems Science*, *52*(14), 2904–2917.

Nekouie, N., & Yaghoobi, M. (2016). A new method in multimodal optimization based on firefly algorithm. *Artificial Intelligence Review*, *46*(2), 267–287.

Orujpour, M., Feizi-Derakhshi, M.-R., & Rahkar-Farshi, T. (2020). Multi-modal forest optimization algorithm. *Neural Computing and Applications*, *32*(10), 6159–6173.

Pan, A., Shen, B., & Wang, L. (2022). Ensemble of resource allocation strategies in decision and objective spaces for multiobjective optimization. *Information Sciences*, *605*, 393–412.

Precup, R.-E., Hedrea, E.-L., Roman, R.-C., Petriu, E. M., Szedlak-Stinean, A.-I., & Bojan-Dragos, C.-A. (2021a). Experiment-based approach to teach optimization techniques. *IEEE Transactions on Education*, *64*, 88–94.

Precup, R.-E., Hedrea, E.-L., Roman, R.-C., Petriu, E. M., Szedlak-Stinean, A.-I., & Bojan-Dragos, C.-A. (2021b). GWO-based optimal tuning of type-1 and type-2 fuzzy controllers for electromagnetic actuated clutch systems. *Ifac Papersonline*, *54*, 189–194.

Qu, B. Y., Liang, J. J., Suganthan, P. N., & Chen, Q. (2014). *Problem definitions and evaluation criteria for the CEC 2015 competition on single objective multi-niche optimization*. Zhengzhou University, Zhengzhou, China: Computational Intelligence Laboratory.

Qu, B. Y., Suganthan, P. N., & Das, S. (2013). A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, *17*(3), 387–402.

Qu, B. Y., Suganthan, P. N., & Liang, J. J. (2012). Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, *16*(5), 601–614.

Rim, C., Piao, S., Li, G., & Pak, U. (2018). A niching chaos optimization algorithm for multimodal optimization. *Soft Computing*, *22*(2), 621–633.

Rokbani, N., Mirjalili, S., Slim, M., & Alimi, A. M. (2022). A beta salp swarm algorithm meta-heuristic for inverse kinematics and optimization. *Applied Intelligence*, *52*(9), 10493–10518.

Singh, D., & Shukla, A. (2022). Manifold optimization with MMSE hybrid precoder for mm-wave massive MIMO communication. *Romanian Journal of Information Science and Technology*, *25*, 36–46.

Song, S., Ji, J., Chen, X., Gao, S., Tang, Z., & Todo, Y. (2018). Adoption of an improved PSO to explore a compound multi-objective energy function in protein structure prediction. *Applied Soft Computing*, *72*, 539–551.

Tan, G. W.-H., Ooi, K.-B., Leong, L.-Y., & Lin, B. (2014). Predicting the drivers of behavioral intention to use mobile learning: A hybrid SEM-Neural Networks approach. *Computers in Human Behavior*, *36*, 332–347.

Tanabe, R., & Ishibuchi, H. (2019). A niching indicator-based multi-modal many-objective optimizer. *Swarm and Evolutionary Computation*, *49*, 134–146.

Wang, R., Hao, K., Huang, B., & Zhu, X. (2023). Adaptive niching particle swarm optimization with local search for multimodal optimization. *Applied Soft Computing*, *133*.

Wang, J., Liang, G., & Zhang, J. (2019). Cooperative differential evolution framework for constrained multiobjective optimization. *IEEE Transactions on Cybernetics*, *49*(6), 2060–2072.

Wang, H., Yang, S., Ip, W. H., & Wang, D. (2012). A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems. *International Journal of Systems Science*, *43*(7), 1268–1283.

Wang, Z.-J., Zhou, Y.-R., & Zhang, J. (2022). Adaptive estimation distribution distributed differential evolution for multimodal optimization problems. *IEEE Transactions on Cybernetics*, *52*(7), 6059–6070.

Xiong, S., Gong, W., & Wang, K. (2023). An adaptive neighborhood-based speciation differential evolution for multimodal optimization. *Expert Systems with Applications*, *211*.

Xue, J., & Shen, B. (2020). A novel swarm intelligence optimization approach: sparrow search algorithm. *Systems Science and Control Engineering*, *8*(1), 22–34.

Yoo, C.-H. (2018). A new multi-modal optimization approach and its application to the design of electric machines. *IEEE Transactions on Magnetics*, *54*(3).

Zamfirache, I. A., Precup, P., & Roman, R.-C. (2023). Neural network-based control using actor-critic reinforcement learning and grey wolf optimizer with experimental servo system validation. *Expert Systems with Applications*, *225*.

Zeng, B., Li, X., Gao, L., Zhang, Y., & Dong, H. (2020). Whale swarm algorithm with the mechanism of identifying and escaping from extreme points for multimodal function optimization. *Neural Computing and Applications*, *32*(9), 5071–5091.

Zhang, J., huang, D.-S. H., Lok, T.-M., & Lyu, M. R. (2006). A novel adaptive sequential niche technique for multimodal function optimization. *Neurocomputing*, *69*(16–18), 2396–2401.

Zou, J., Deng, Q., Zheng, J., & Yang, S. (2020). A close neighbor mobility method using particle swarm optimizer for solving multimodal optimization problems. *Information Sciences*, *519*, 332–347.