SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2016, Vol. 59 052101:1–052101:13
doi: 10.1007/s11432-015-5372-0

Rotated neighbor learning-based auto-configured evolutionary algorithm

Yuanjun LAILI, Lin ZHANG*, Fei TAO & Pingchuan MA

School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191*, China*

Received February 26, 2015; accepted March 30, 2015; published online January 20, 2016

Abstract More and more evolutionary operators have been integrated and manually configured together to solve a wider range of problems. Considering the very limited progress made on the automatic configuration of evolutionary algorithms (EAs), a rotated neighbor learning-based auto-configured evolutionary algorithm (RNL-ACEA) is presented. In this framework, multiple EAs are combined as candidates and automatically screened for different scenarios with a rotated neighbor structure. According to a ranking record and a group of constraints, the algorithms can be better scheduled to improve the searching efficiency and accelerate the searching pace. Experimental studies based on 14 classical EAs and 22 typical benchmark problems demonstrate that RNL-ACEA outperforms other six representative auto-adaptive EAs and has high scalability and robustness in solving different kinds of numerical optimization problems.

Keywords multiple evolutionary algorithms, algorithm auto-configuration, rotated neighbor structure, hyper-heuristic, numerical optimization

Citation Laili Y J, Zhang L, Tao F, et al. Rotated neighbor learning-based auto-configured evolutionary algorithm. Sci China Inf Sci, 2016, 59(5): 052101, doi: 10.1007/s11432-015-5372-0

1 Introduction

Evolutionary algorithm (EA) [1–3], which takes advantage of bio-inspired operators to do population-based iterative evolution within reasonable time, has been proved to be one of the most efficient algorithms for solving non-deterministic Polynomial-time hard (NP-hard) problems [4–6]. With frequently changing environment, hundreds of improved and hybrid EAs have been presented [7–9]. However, according to "no free lunch" theory, when the nature of optimization problem varies, an particularly designed algorithm will easily become inefficient or even fail and a cumbersome redesign work is then required. To settle this issue, hyper-heuristic has been presented to collect under-layer heuristics and dynamically select them for solving more problems [10]. The main focus is the design of high-level rule [11,12]. It tries to choose or combine several heuristics in a specific stage to solve a problem [13]. The objects to be selected by hyper-heuristic are generally problem-dependent priori knowledge.

Similarly, memetic algorithm is also established as a framework, especially for integrating several local search strategies (also called as memes) in evolutionary iteration. In the early stage, single local search and genetic updating operators are hybridized for specific problems [14]. After years of development, the

* Corresponding author (email: zhanglin@buaa.edu.cn)

idea of multi-memes is presented [15,16]. In each generation, local search strategy (i.e., meme) is selected by Meta-Lamarckian learning strategy to update individuals.

Clearly, only local search strategies and simple priori knowledge are selected as candidates in the above frameworks. For extending the selection framework to evolutionary operators, multi-method search [17,18] and dynamic configuration method [19] are proposed with general evolutionary procedure. In multi-method search, the whole EA is adopted as candidates. In dynamic configuration method, both algorithms and their operators are flexibly combined as modules. Except that, for solving conflicts existing in many engineering circumstances, we may require multiple decision makers in different aspects. In this aspect, some negotiation and preference schemes are also proposed to handle complex optimization [20,21].

However, unfortunately, few mechanisms have been well designed for choosing suitable EAs in both multi-method search and dynamic configuration framework. Also, the complexity of such multiple EAs scheme is decided by both the upper-layer selection mechanism and the candidates with the largest complexity. With the growth of problem solution space and the additional algorithm switching mechanism, the whole searching time will thus be exponentially increased. We have reason to believe that the selection mechanism for generating a combined EAs scheme with lower time complexity is still an urgent issue to be explored.

In this paper, a new multiple EAs scheme, named rotated neighbor learning-based auto-configured evolutionary algorithm (RNL-ACEA), is proposed. Instead of original selection mechanism applied in multi-memes and multi-method search, the algorithm used in each individual is updated according to the algorithm rank records of both its neighbor and itself. The dynamic changing structure is expected to improve the collaboration between individuals and find better algorithms with less time. We introduce up to 14 kinds of classical EAs as the basis of our multi-EAs scheme. To demonstrate the performance of RNL-ACEA, experiments and discussions are established on 22 complex benchmark functions with different dimensions. The comparison analysis against six state-of-the-art selection mechanisms from both multi-method schemes and memetic algorithms and the parameter tuning of our method are all included.

This paper is structured in the following way. Section 2 gives the review of state-of-the-art memetic algorithms, multi-method search, and configuration method of EA. Section 3 provides the workflow and detailed explanation of RNL-ACEA. Experimental studies between the new proposed RNL-ACEA and the state-of-the-art selection mechanisms with multiple EAs scheme are conducted in Section 4. The conclusion of this paper and some potential future work are drawn in Section 5.

2 Literature review

Developed from the hybridization of single local search operator and genetic one, memetic algorithms have gone through three generations [22][1)]. Under this circumstance, Ong et al. [16] have classified the most typical selection mechanisms used in memetic algorithms. They include simple random method, simple inheritance mechanism, subproblem decomposition strategy, and so on. In this scheme, some classical hill-climbing strategies early elaborated by Schwefel [23] are also applied as local search memes and listed in [15,16].

From the algorithm design point of view, Nguyen et al. [24] have tested and deeply analyzed the general parameter settings of local search procedure during the searching process. Le et al. [25] have introduced the influence of the local optimum structure and connectivity structure in memetic algorithms. These parameters and structures are proved to be very critical in design of a successful memetic algorithm [26]. From the parallelization point of view, the diversity-based static and dynamic adaptive strategies, that is, diversity-based static adaptive strategy (PMA-SLS) and diversity-based dynamic adaptive strategy (PMA-DLS) are designed, in which the number of individuals to perform local search is calculated according to Gaussian distribution and online entropy ratio, respectively [27].

1) Memetic algorithm. http://en.wikipedia.org/wiki/Memetic_algorithm.

Except the research on the structure of memetic algorithm, many efforts have been put on the combination of local search memes and other EAs. Typical examples are the particle swarm optimization-based memetic algorithm [28], differential evolution-based memetic algorithm [29], memetic computation on immune systems [30], and so forth.

For further extending its application, researchers have established a new branch of evolutionary computing, called memetic computing. Representative researches on it include the establishment of coevolving memetic scheme [31], the multi-stage memetic exploration [32], and other heuristics as memes [33] and so on. Most of them try to combine more heuristics for solving wider range of problems like the development of hyper-heuristic. Clearly, the research of extensive memetic computing is still at an early stage.

Efforts put on the combined use of multiple evolutionary operators like memes are less. Multi-method search first presented in 2007 [17] is a typical one. Several evolutionary operators are combined and their operating scopes are chosen and defined in advance. After that, Vrugt et al. [18] try to improve the scheme using some acquired information to update the number of individuals handled by each operator, so as to make the searching process dynamic. Obviously, operator selection mechanism is still lacking.

Recently, Hadka et al. [34,35] presented a Borg multi-objective EA by applying the above multi-method search and a biased probability-based selection mechanism for auto-adaptation. Grobler et al. [36] established a framework using EAs as low-level heuristics and built an upper-layer heuristic for selecting them. By digging the population trait, Peng et al. [37] proposed a group-based portfolio based on multi-method search. Except that, the research on the selection mechanism for such multi-method scheme can also be found in the study of adaptive differential evolution, in which multiple differential mutation strategies are applied as candidates [38,39].

The multi-method search has been used in many complex optimization problems in recent times, such as [40,41]. From the operator point of view, the candidates can be generally applied to wider range of problems than the memes in memetic algorithm.

Dynamic configuration scheme [19] is designed especially for full use of evolutionary operators and the improvement strategies. Different from multi-method search, it focuses mainly on the combination ways of operators in the evolutionary procedure. Tao et al. [19] have divided the configuration into three layers: parameter configuration, operator configuration, and algorithm configuration.

Parameter configuration refers to general parameter adaptation during the iterations. Operator configuration means to combine the existing operators from different algorithms to generate new evolutionary scheme in a specific period. Algorithm configuration is defined as applying diverse algorithms into different stages of the process, as well as multi-method search. Similarly, the specific adaptive configuration mechanism has not been addressed yet.

Although many studies are gradually carried out on the design of multiple EAs scheme, automatic selection mechanism, especially for evolutionary operators has yet to be well studied and explored. As a result, with those dynamic configuration ways, the existing EAs are still far from being efficiently used and properly assembled.

3 Rotated neighbor learning-based auto-configured evolutionary algorithm

In this section, we will give the basic framework of RNL-ACEA. The specific neighbor learning mechanism and its rotated structure are elaborated following the framework, respectively. At this level, we will explain how could it adapt for different problems without increasing searching time.

In RNL-ACEA, the rank record presented by Burke et al. [42] is applied for evolutionary record. With a rank list, the high-ranked algorithm candidates for each individual can be used to guide their neighbor and enhance the collaboration between individuals. As shown in the pseudocode of Algorithm 1, the proposed RNL-ACEA includes mainly four parts.

(1) Initialization (Steps 1–3): The whole population is generated with random Gaussian distribution. We define evolutionary flag, that is, $\{\text{Count}_i | i \in [1, N]\}$ as the number of generations after the last improvement of each individual and $\boldsymbol{rk} = \{rk_{i,j} | i \in [1, N], j \in [1, M]\}$ as the rank matrix, in which $rk_{i,j}$

represents the rank of candidate j for individual i. In the beginning, N algorithms are randomly selected from M algorithm candidates and denoted as $\boldsymbol{A} = \{A_i | i \in [1, N]\}$ for generating new individuals. An algorithm in the candidate pool can be repeatedly selected. Therefore, the size of the candidate pool M can be either larger or smaller than N.

(2) Evolution (Steps 4–7): The selected algorithms

$$\boldsymbol{A} = \{A_i | i \in [1, N]\}$$

are performed on different individuals to find new positions in the solution space.

(3) State record (Steps 8–18): Both the individual states and the algorithm rank records are refreshed in turn after the evolution.

(4) Rotated interaction (Steps 20–31): Neighbor finding and neighbor learning make the individual exchange for good algorithm and good information.

Algorithm 1 Rotated neighbor learning-based auto-configured evolutionary algorithm (RNL-ACEA)

1: Generate an initial population with N individuals
2: Set $\text{Count}_1 = \cdots = \text{Count}_N = 0$ and $rk_{1,1} = \cdots = rk_{i,j} = \cdots = rk_{N,M}$, $i \in [1, N]$, $j \in [1, M]$
3: Randomly select N algorithms from M candidates
4: **while** Stopping conditions not satisfied **do**
5: **for** each individual i **do**
6: Perform the selected algorithm A_i to generate a new offspring
7: Update the individual by Elitist strategy
8: **if** the fitness value of the individual is improved **then**
9: **if** $r_{i,A_i} < rk_{\max}$ **then**
10: $r_{i,A_i} += 1$
11: **end if**
12: $\text{Count}_i = 0$
13: **else**
14: $\text{Count}_i += 1$
15: **if** $r_{i,A_i} > 0$ **then**
16: $r_{i,A_i} -= 1$
17: **end if**
18: **end if**
19: **end for**
20: **for** each individual i **do**
21: Obtain the serial number of the neighbor n_i according to the rotated structure
22: **if** the updated fitness value of the neighbor is better than i and $\text{Count}_i > \text{CBOUND}$ **then**
23: Select the candidate of neighbor with the highest rank $\max(rk_{\text{neighbor},j})$ as A_i
24: **else**
25: **if** $\text{Count}_i < \text{CBOUND}$ **then**
26: Select the candidate of i with the highest rank $rk_{i,j}$ as A_i
27: **end if**
28: **else**
29: Randomly select a candidate as A_i
30: **end if**
31: **end for**
32: **end while**
33: Output the global best individual

3.1 Basic neighbor learning mechanism

Good algorithms can lead to fast evolution, while bad ones may slow down the evolutionary pace. Gene bit interaction between individuals can only guide the search, but not speed up the evolution within group. During the process, if bad algorithms exist in the group, the individuals with slower evolving speed will lag the whole searching process. It is a waste of time and resources. So, the main target of our method is to find suitable algorithms for a specific problem and eliminate the bad ones from the process dynamically.

As we analyzed in Section 2, there are three main categories for upper-layer algorithm selection mechanism: (1) random strategy for evenly applying all candidates, (2) rule-based control with priori information and some choice function, and (3) performance-based control with history record. Considering the rank records with the evolutionary flags, the basic neighbor learning designed in this paper belongs mainly to performance-based control.

During the search, individuals are interactively and randomly located in the solution space. The rank records for one individual are also applicable for others. Through the comparisons of the current updated fitness value with its neighbor, each individual can use the information of the neighbor to guide its own searching direction.

Specifically, under a connect structure, there are three cases for each individual to select algorithm for the next generation.

Case 1: the individual i is evolving, that is, the evolutionary flag is within CBOUND. In this scenario, it can keep its pace and the candidate with the highest rank $\max(rk_{i,j})$ for individual i will be selected.

Case 2: the individual i is not evolving for over CBOUND generations and its neighbor holds a better fitness value. In this case, the candidate with the highest rank $\max(rk_{\text{neighbor},j})$ for its neighbor will be introduced.

Case 3: the individual i is not evolving for over CBOUND generations and its neighbor is not better than it. That indicates a convergence trend of the population. To increase the searching diversity, a randomly selected algorithm will be applied for the next generation.

The random selection step after the neighbor learning is mainly designed to eliminate bad algorithms and try to keep the population updating with different candidates as far as possible.

In the neighbor learning mechanism, CBOUND is an important parameter. It is used to balance the diversity of algorithm and the collaboration of population. If CBOUND is too big, the individual can get very little information from the neighbor records. The collaboration pace will be largely slowed down and each individual will perform independent searching and algorithm selection. Simultaneously, the random strategy will have little chance to work, especially at the early stage of the evolutionary searching. On the other hand, if CBOUND is too small, the algorithm selection will be decided largely by the random strategy but not the history records.

Hence, if the random strategy and the learning strategy are well balanced, the candidates will evenly perform on the early stage searching and the good ones will be selected after a certain time of individual collaboration. Even if early convergences of both algorithm and population happen, the random algorithm selection and the population refreshment will both help to renew them within limited generations.

3.2 Rotated structure and neighbor finding

In the neighbor learning mechanism, the most crucial problem is how to choose the neighbor during the interaction. The connection structure of individuals has been researched for years to establish efficient parallel EAs. The most commonly used topologies are ring topology [43], grid topology [44], full mesh topology [45], and so on. Matsumura et al. [46] compared and analyzed the influences of different topologies and found the ring structure performs the best in keeping population diversity. Apparently, it can also be applied to our learning selection scheme. Via one-way ring structure, one neighbor connection can be assured in the workflow of RNL-ACEA as well.

However, with fixed neighbor assigned beforehand, it requires $N-1$ steps at least to transmit good information to the whole population. As mentioned by Matsumura et al. [46], the convergence rate of ring structure is very slow. The learning object of an individual is always the same. It has no chance to learn from other individuals at all. For improving the learning diversity and enhancing the information transmission, we propose a rotated structure, as shown in Figure 1.

In the beginning, the structure is totally the same with ring topology. Let n_i be the neighbor of the ith individual. The initial neighbor of i is calculated as $n_i = g_i + 1$. After that, the neighbor can be calculated by (1) within each generation:

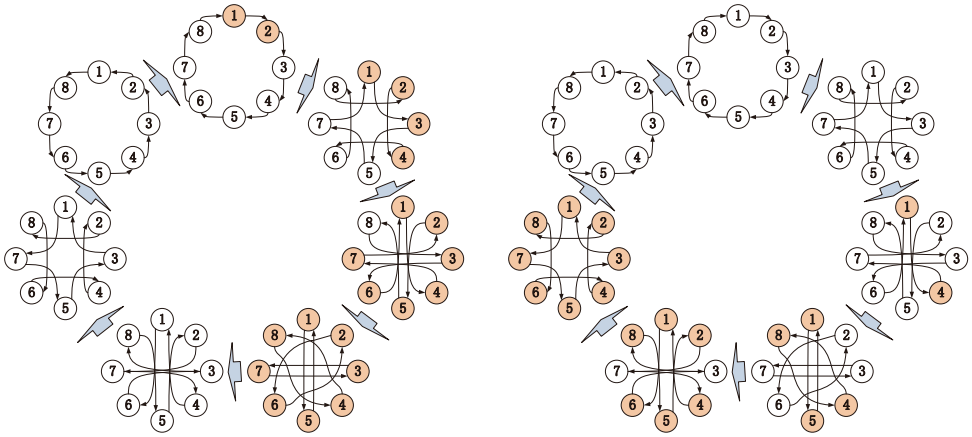

Figure 1 (Color online) Rotated structure for neighbor learning in RNL-ACEA.

$$n_i = i + 1 + \text{gen} \mod (N - 1). \tag{1}$$

In this equation, gen represents the current generation in evolution. The only difference between the rotated structure and the ring topology is that the neighbor is changed in every generation.

Take a population with eight individuals as an example, as shown in Figure 1, if the selected algorithm of individual 1 performs very well, then it will pass the candidate to 2 at the first period. Next, individual 1 and individual 2 will send the candidate number to their next neighbor, that is, 3 and 4, respectively, if both of them are evolved in the previous generation. Clearly, it requires at most four steps to transmit the good one to the whole population. In general, the information transmission steps will be reduced from $\log N$ at least to $N - 1$ at most. Meanwhile, the collaborations between individuals in total do not increase any more. Each one has equal opportunity to learn from all of the other individuals during the iterations.

The neighbor learning process is much simpler than other rule-based or performance-based selection mechanisms with some choice functions. During the iterative search, the individuals learn from each other and find better algorithms by rotated interaction. With rotated neighbor learning, individuals will have more chance to get suitable algorithms. The diversity of both population and algorithm can also be well maintained by the combination use of its supplementary random strategy and evolutionary flags.

4 Experiments and discussion

In this section, we apply our proposed RNL-ACEA on 22 typical benchmark problems. Eight of them are selected from CEC'2005 benchmarks and the others are typical complex numerical functions. Both unimodal and multimodal functions are tested with dimensions 30, 50 and 100 in the experiments. First, 14 typical EAs are adopted as the under-layer candidates. The performance of rotated structure in RNL-ACEA based on them is analyzed. Then, to demonstrate the efficiency of RNL-ACEA with the state-of-the-art mechanisms, six kinds of hyper-heuristics (as the upper-layer selection methods) are reimplemented in the experiments. Besides, for verifying the scalability and robustness of RNL-ACEA, the numerical functions with 1000 dimensions are tested. At last, the key parameter of RNL-ACEA is tuned and analyzed to show its impact on the design and application of RNL-ACEA for a wider range of continuous numerical problems.

4.1 Experiment settings

(1) Algorithms for comparison

First, we set the auto-configured EA with simple ring structure-based neighbor learning as NL-ACEA, and the rotated neighbor learning-based one as RNL-ACEA. The above-mentioned six typical hyper-

Table 1 The six hyper-heuristics for comparison

Abbreviation	Algorithm	Abbreviation	Algorithm
AMALGAM	AMALGAM-SO presented in [18]	Inheritance	Simple inheritance mechanism introduced in [16]
Borg	Borg mechanism presented in [35]	Biased-RW	Biased roulette wheel strategy introduced in [16]
Tabu	Tabu strategy presented in [43]	Subpro-Dec	Subproblem decomposition strategy introduced in [16]

Table 2 The algorithm candidates applied in the experiments

No.	Algorithm	No.	Algorithm
1	Genetic algorithm with single point crossover/mutation	8	Harmony search [49]
2	Particle swarm optimization	9	Differential evolution with best/2 mutation [39]
3	Differential evolution with rand/1 mutation [39]	10	Cuckoo search [50]
4	Iterative local search [47]	11	Differential evolution with target-to-best/1 mutation [39]
5	Differential evolution with best/1 mutation [39]	12	Variable neighborhood search [51]
6	Artificial bee colony algorithm [48]	13	Greedy randomized adaptive search procedure [52]
7	Differential evolution with rand/2 mutation [39]	14	Continuous ant colony optimization [53]

Table 3 The initial parameter settings of the algorithm candidates

No. of algorithms	Parameters	Settings
1	Crossover probability (Pc), Mutation probability (Pm)	0.8, 0.15
2	Cognitive factor ($c1$), Social factor ($c2$), Inertia weight (w)	2.0, 2.0, 0.85
3, 5, 7, 9, 11	Weighting factor (F), Crossover constant (cr)	0.8, 0.3
4, 12, 13	Local search strategy	Bit climbing
6	The trial limitation of scout bee	50
8	The choosing rate of harmony memory, the choosing rate of neighboring value	0.9, 0.3
14	The size of the archive, locality of the search process, speed of convergence	10, 1×10^{-4}, 0.85

Table 4 The 14 test functions for testing the proposed algorithm

No.	Func name	Range	Optimum	Accuracy	No.	Func name	Range	Optimum	Accuracy
T1	Sphere	$[-100,100]^D$	0	1×10^{-6}	T8	NoiseQuadric	$[-100,100]^D$	0	1×10^{-2}
T2	Schwefel P2.22	$[-100,100]^D$	0	1×10^{-6}	T9	RotatedF6	$[-100,100]^D$	0	1×10^{-1}
T3	Schwefel P1.2	$[-100,100]^D$	0	1×10^{-6}	T10	HighElliptic	$[-100,100]^D$	0	1×10^{-6}
T4	Schwefel P2,21	$[-100,100]^D$	0	1×10^{-6}	T11	2Dminima	$[-5,5]^D$	0	1×10^{-6}
T5	Rosenbrock	$[-5,100]^D$	0	1×10^{-2}	T12	Tablet	$[-100,100]^D$	0	1×10^{-6}
T6	Step	$[-100,100]^D$	0	1×10^{-6}	T13	Diff power	$[-100,100]^D$	0	1×10^{-6}
T7	Ellipse	$[-100,100]^D$	0	1×10^{-6}	T14	Rastrigin	$[-5,5]^D$	0	1×10^{-6}

heuristics and their abbreviations are listed in Table 1. Among them, Borg, Tabu, Subprob-Dec, and Biased-RW all belong to the category of performance-based control with history record. The under-layer algorithm candidates are provided in Table 2 with serial numbers. Then, the initial settings of these algorithms are given in Table 3. These parameters will not be changed during the whole experiments.

(2) Benchmarks

We apply 14 complex functions used in [54] and 8 shifted benchmarks listed in CEC'2005 [55] to test the above algorithms. The search range and accuracy of the 14 test functions are listed in Table 4. The details of the eight benchmarks from CEC'2005 can be found in [55]. In the following experiments, we test the algorithms on the 14 test functions with dimensions 50 and 100 and eight benchmarks on dimensions 30 and 50, respectively.

(3) Performance metric

Following the evaluation criteria in [55,56], all of the experiments are taken 30 runs to get the average value. And we mainly evaluate the algorithm using Err, SR, and TIME.

- Err: The average error of the optimal solution and the best fitness value found in the 30 runs.
- SR: Success rate of the 30 runs.
- TIME: The average searching time in the 30 runs. Its unit is second (s).

Table 5 t-test comparison of NL-ACEA and RNL-ACEA on small- and large-scale tests

Small scale	ERR	SR	TIME	Large scale	ERR	SR	TIME
t-test	0.3255	0.1498	0.03827	t-test	0.3283	0.1479	0.0294
Wilcoxon-test	0.0431	0.0679	0	Wilcoxon-test	0.1380	0.1088	0.0009

Table 6 t-test comparison of Err between RNL-ACEA and other six selection mechanisms

RNL-ACEA v.s.	t	Unadjusted p	Bonferroni p	r_i	Holm p	Hochberg p
AMALGAM	1.163	0.2512	> 1	> 1	> 1	> 1
Borg	1.084	0.2845	> 1	> 1	> 1	> 1
Tabu	1.090	0.2828	> 1	> 1	> 1	> 1
Inheritance	1.448	0.1547	0.9282	0.4641	0.4641	0.4641
Subpro-Dec	2.355	0.0232	0.1392	0.0464	0.0464	0.0464
Biased-RW	2.198	0.0334	0.2004	0.0334	0.0334	0.0334

Table 7 t-test comparison of SR between RNL-ACEA and other six selection mechanisms

RNL-ACEA v.s.	t	Unadjusted p	Bonferroni p	r_i	Holm p	Hochberg p
AMALGAM	3.366	0.0016	0.0096	0.0096	0.0096	0.0096
Borg	2.799	0.0076	0.0456	0.0380	0.0380	0.0380
Tabu	4.863	0	0	0	0	0
Inheritance	5.560	0	0	0	0	0
Subpro-Dec	10.355	0	0	0	0	0
Biased-RW	8.670	0	0	0	0	0

Also, the population size in the experiments is set to be 20 and the maximum number of generations $G_{\max}$ is set to be $N \times 10000$, in which N represents the number of variables. In the comparison, the parameter of RNL-ACEA, CBOUND and $rk_{\max}$ are initially set to be 10 and 100, respectively. The stopping criterion of the whole iteration is that if the accuracy is arrived or the number of generations has arrived to the maximum number $G_{\max}$. The level of significance used for statistical analysis [54,56] in this paper is uniformly set as 0.05.

4.2 Comparison of NL-ACEA and RNL-ACEA

Pair comparisons between NL-ACEA and RNL-ACEA are carried out using t-test. Table 5 shows the p-values of Err, SR, and TIME between NL-ACEA and RNL-ACEA in different statistical tests. From the t-test comparison, it can be observed that the rotated structure significantly decreases the searching time in both small- and large-scale tests. The p-value of TIME becomes smaller as the problem scale increases. On the contrary, the rotated structure on searching precision and success rate are insignificant. However, from the results on shifted benchmarks, especially Shifted Rotated Weierstrass Function and Schwefel's Problem 2.13, we can find that the average success rate and precision are both decreased without neighbor rotation.

4.3 Comparison of RNL-ACEA and other six hyper-heuristics

(1) Solution quality

Considering the precision and success rate independently, the results of t-test pair comparison on Err and SR between RNL-ACEA and other six hyper-heuristics are in Tables 6 and 7. The p-values obtained by Bonferroni-Dunn's, Holm's, and Hochberg's procedure are also provided.

Except the pair comparison, non-parametric Friedman's test is also employed to evaluate RNL-ACEA and other six hyper-heuristics on the selection of EAs. The rankings in terms of Err and SR are listed in Table 8. The approximate ranking of the seven methods from the solution quality point of view can be summarized as RNLACEA > Borg > AMALGAM > Tabu > Inheritance > BiasedRW > SubproDec.

(2) Searching time

From the perspective of searching time, we also employ the pair comparisons using t-test to analyze

Table 8 Friedman-test comparison between RNL-ACEA and other six selection mechanisms

Small scale	Err	SR	TIME	Large scale	Err	SR	TIME
RNL-ACEA	2.23	5.52	1.55	RNL-ACEA	2.05	5.77	1.23
AMALGAM	2.86	4.89	4.09	AMALGAM	2.84	4.73	4.09
Borg	2.66	5.30	3.09	Borg	2.66	4.77	3.50
Tabu	3.82	4.11	3.59	Tabu	4.23	4.09	3.91
Inheritance	4.25	3.59	5.14	Inheritance	3.77	3.82	4.91
Subpro-Dec	6.32	1.98	6.05	Subpro-Dec	6.23	2.05	6.23
Biased-RW	5.86	2.61	4.50	Biased-RW	6.23	2.77	4.14
χ^2	86.661	75.266	60.117	χ^2	96.064	67.350	64.870
p	0	0	0	p	0	0	0

Table 9 t-test comparison of TIME between RNL-ACEA and other six selection mechanisms

RNL-ACEA v.s.	t	Unadjusted p	Bonferroni p	r_i	Holm p	Hochberg p
AMALGAM	2.860	0.0065	0.0391	0.0391	0.0391	0.0391
Borg	2.911	0.0057	0.0341	0.0284	0.0284	0.0284
Tabu	2.826	0.0071	0.0427	0.0285	0.0285	0.0285
Inheritance	2.820	0.0072	0.0435	0.0217	0.0217	0.0217
Subpro-Dec	3.138	0.0031	0.0184	0.0061	0.0061	0.0061
Biased-RW	3.411	0.0014	0.0085	0.0014	0.0014	0.0014

the efficiency of RNL-ACEA. It can be observed from Table 9 that RNL-ACEA outperforms other six methods on the aspect of TIME. The p-values obtained from Bonferroni-Dunn's, Holm's, and Hochberg's procedure are all under the level of significance.

On the other hand, the results of Friedman's test shown in Table 8 also provide the rankings of these seven methods in terms of TIME. The efficiency ranking can be concluded as RNLACEA < Borg < Tabu < AMALGAM < BiasedRW < Inheritance < SubproDec.

On the whole, RNL-ACEA with rotated neighborhood structure and learning strategy established based on a group of rank records and evolutionary flags can find suitable candidates faster for all of the 22 functions. The diversities and collaborations of both population and algorithm candidates are well maintained, thus the solution quality and searching time are comprehensively improved.

4.4 Scalability and robustness of RNL-ACEA

(1) Scalability

To verify the scalability of RNL-ACEA in complex numerical optimizations, experiments on 14 test functions with dimension 1000 are carried out. Accorting to Table 10, it is clear that even if the problem dimension increased to 1000, RNL-ACEA can still find the optimal solution with certain precision. However, the searching times are increased accordingly. Although the upper-layer selection mechanism has low time complexity, how to make the under-layer evolution converge faster and find the optimal solution simultaneously is still a big challenge.

(2) Robustness

As we introduced in Section 3, RNL-ACEA has only two parameters, that is, CBOUND and $rk_{\max}$. CBOUND is used to control the algorithm learning between individuals, while $rk_{\max}$ is applied to avoid biased algorithm selection. To assess the robustness of RNL-ACEA, parameter-tuning experiments are also carried out in this paper. With fixed $rk_{\max} = 100$, we test RNL-ACEA with different CBOUND values on the 14 test functions, respectively. When CBOUND changes from 5 to 50, the precision and success rate can both be well maintained. Nevertheless, when it continuously increases, the success rate and precision are decreased. The same trend can also be obtained on the aspect of searching time.

Considering the insignificant changes on precision and success rate, statistical analysis is only taken on TIME by t-test pair comparison and Friedman's test multiple comparison. The p-values obtained by t-test comparison of RNL-ACEA are shown in Table 11, and the rankings obtained by Friedman's

Table 10 Result of RNL-ACEA for 14 test functions with dimension 100

T1	ErrSR	TIME	T2	ErrSR	TIME	T3	ErrSR	TIME	T4	ErrSR	TIME	T5	ErrSR	TIME	T6	ErrSR	TIME	T7	ErrSR	TIME
0	1	173.123	0	1	260.245	0	1	4941.86	0	1	1434.45	0	1	3070.41	0	1	173.868	0	1	592.627
T8	ErrSR	TIME	T9	ErrSR	TIME	T10	ErrSR	TIME	T11	ErrSR	TIME	T12	ErrSR	TIME	T13	ErrSR	TIME	T14	ErrSR	TIME
0	1	1131.43	0	1	702.405	0	1	199.946	0	1	2062.32	0	1	50.8057	0	1	921.716	0	1	301.069

Table 11 t-test comparison of RNL-ACEA under different parameter settings of CBOUND

CBOUND	5-10	5-20	5-30	5-50	5-100	10-20	10-30	10-50	10-100	20-30	20-50	20-100	30-50	30-100	50-100
p-value	0.0165	0.0229	0.0206	0.6794	0.0047	0.7202	0.4635	0.0004	0.0018	0.2856	0.0003	0.0022	0	0.0017	0.0062

Table 12 Friedman-test comparison of RNL-ACEA under different parameter settings of CBOUND

CBOUND	5	10	20	30	50	100	χ^2	p
Rank	4.14	2.21	2.64	1.43	4.71	5.86	56.490	0

Table 13 t-test comparison of RNL-ACEA under different parameter settings of $rk_{\max}$

$rk_{\max}$	5-10	5-20	5-30	5-50	5-80	5-100	5-500	10-20	10-30	10-50	10-80	10-100	10-500	20-30
p-value	0.0562	0.0127	0.0142	0.0128	0.0219	0.0361	0.2101	0.0019	0.0072	0.0156	0.0374	0.0998	0.1014	0.0364
$rk_{\max}$	20-50	20-80	20-100	20-500	30-50	30-80	30-100	30-500	50-80	50-100	50-500	80-100	80-500	100-500
p-value	0.0478	0.0979	0.2470	0.0544	0.2023	0.5121	0.6837	0.0291	0.1264	0.0107	0.0147	0.0132	0.0186	0.0223

Table 14 Friedman-test comparison of RNL-ACEA under different parameter settings of $rk_{\max}$

$rk_{\max}$	5	10	20	30	50	80	100	500	χ^2	p
rank	6.57	5.79	4.43	2.64	1.86	2.79	4.86	7.07	60.810	0

test comparison are summarized in Table 12. Obviously, when CBOUND is in the range of [10,30], the searching time of RNL-ACEA changed little. Likewise, Friedman's test is also introduced to evaluate its searching time with different settings of CBOUND. The ranking results can be shown in Table 11. We can observe that when CBOUND is set as 30, RNL-ACEA performs the best.

With fixed CBOUND = 10, we set different $rk_{\max}$ on 14 test functions as well. We found that no matter how $rk_{\max}$ changed, the precision and success rate of RNL-ACEA can be well maintained. From Tables 13 and 14, it is clear that when $rk_{\max}$ changes from 30 to 80, the change of TIME is insignificant. When it changes from 5 to 50, the searching time of RNL-ACEA is gradually reduced. When it increased further (from 50 to 500), the searching time of RNL-ACEA is bounced back.

In theory, small $rk_{\max}$ can lead to random algorithm selection. Algorithms with good performance can always get the rank equal to $rk_{\max}$. Then, the selection probability of each algorithm will become even. On the contrary, large $rk_{\max}$ can result in biased search because when the rank of an algorithm becomes very large, it is hard to be reduced to a low level. So, it can be concluded that when $rk_{\max}$ changes from 30 to 80, the robustness of RNL-ACEA can be well held. When it is set as 50, RNL-ACEA performs the best in solving those continuous benchmark problems.

As CBOUND and $rk_{\max}$ both have little influence on the whole performance of search when located in a suitable range, it can be seen that RNL-ACEA is easy to implement with lesser parameters and high adaptation.

5 Conclusion

In this paper, we proposed a rotated neighbor learning-based multiple EAs scheme, RNL-ACEA, to schedule suitable algorithms for individuals via rotated information learning. We attempted to use the random replacement strategy in accordance with a bound of evolutionary state to keep the diversity of both algorithms and population. Then, we designed a rotated neighborhood structure to scatter suitable candidates among individuals. Experiments conducted based on 22 classical benchmark problems

demonstrated that RNL-ACEA outperforms the six state-of-the-art schemes from the perspectives of both solution quality and efficiency. And its parameter is easy to tune with high robustness and feasibility in a certain range.

As described in the previous sections, RNL-ACEA can be applied on both continuous numerical problem and combinatorial optimization problem in industrial engineering. For continuous numerical problem, engineers only need to provide the specific fitness evaluation function to recall the algorithm. For combinatorial optimization problem, RNL-ACEA with its operators can also be used by adding an encoding/decoding function in evaluation. For example, to solve the service composition optimal selection problem in Cloud Manufacturing [46], a simple float-to-integer mapping can be adopted as the decoding way in evaluation. All the candidate operators are independent from the specific problem. Therefore, this method can be well applied to scheduling and numerical optimization in many kinds of distributed systems such as service-oriented manufacturing system [57–59], energy management system [60], and so on.

In this scenario, we are motivated to conduct an auto-configured EA library with a uniform problem input interface. By integrating numbers of EAs as under-layer candidates and implementing a group of hyper-heuristic strategies to automatically select EAs, it is expected that efficient optimization of engineering problems can be obtained with less design work, repetition, and manual adjustment.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (NSFC) (Grant Nos. 61374199, 51475032), Beijing Municipal Natural Science Foundation (Grant No. 4142031), National High Technology Research and Development Program of China (863) (Grant No. 2013AA041302) and Innovation Foundation of Beihang University for PhD Graduates (Grant No. YWF-14-YJSY-001).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Osman I H, Kelly J P. Meta-heuristics: an overview. In: Meta-Heuristics. Berlin: Springer, 1996. 1–21
2 Kochenberger G A. Handbook in Metaheuristics. Berlin: Springer, 2003
3 Talbi E G. Metaheuristics: From Design to Implementation. Hoboken: John Wiley & Sons, 2009
4 Paz A, Moran S. Non deterministic polynomial optimization problems and their approximations. Theoretical Comput Sci, 1981, 15: 251–277
5 Yu Y, Yao X, Zhou Z H. On the approximation ability of evolutionary optimization with application to minimum set cover. Artif Intell, 2012, 180–181: 20–33
6 Qian C, Yu Y, Zhou Z H. An analysis on recombination in multi-objective evolutionary optimization. Artif Intell, 2013, 204: 99–119
7 Yang X S. Engineering Optimization: an Introduction With Metaheuristic Applications. Hoboken: John Wiley & Sons, 2010
8 Wang Y, Li B, Yuan B. Hybrid of comprehensive learning particle swarm optimization and SQP algorithm for large scale economic load dispatch optimization of power system. Sci China Inf Sci, 2010, 53: 1566–1573
9 Zhang X J, Guan X M, Hwang I, et al. A hybrid distributed-centralized conflict resolution approach for multi-aircraft based on cooperative co-evolutionary. Sci China Inf Sci, 2013, 56: 128202
10 Burke E K, Kendall G, Newall J, et al. Hyper-heuristics: an emerging direction in modern search technology. In: International Series in Operations Research and Management Science. Dordrecht: Kluwer Academic Publishers, 2003. 457–474
11 Burke E K, Hyde M, Kendall G, et al. A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics. Beilin: Springer, 2010. 449–468
12 Burke E K, McCollum B, Meisels A, et al. A graph-based hyper-heuristic for educational timetabling problems. Eur J Oper Res, 2007, 176: 177–192
13 Qu R, Burke E K. Hybridizations within a graph based hyper-heuristic framework for university timetabling problems. J Oper Res Soc, 2009, 60: 1273–1285
14 Moscato P. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Computation Program, C3P Report, 1989, 826: 1989
15 Ong Y S, Keane A J. Meta-Lamarckian learning in memetic algorithms. IEEE Trans Evolut Comput, 2004, 8: 99–110
16 Ong Y S, Lim M H, Zhu N, et al. Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybernet Part B: Cybernet, 2006, 36: 141–152
17 Vrugt J A, Robinson B A. Improved evolutionary optimization from genetically adaptive multimethod search. Proc National Academy Sci, 2007, 104: 708–711

18 Vrugt J A, Robinson B A, Hyman J M. Self-adaptive multimethod search for global optimization in real-parameter spaces. IEEE Trans Evolut Comput, 2009, 13: 243–259
19 Tao F, Laili Y J, Liu Y, et al. Concept, principle and application of dynamic configuration for intelligent algorithms. IEEE Syst J, 2014, 8: 28–42
20 Bechikh S, Said L B, Ghédira K. Negotiating decision Makers' reference points for group preference-based evolutionary multi-objective optimization. In: Proceedings of the 11th IEEE International Conference on Hybrid Intelligent Systems, Malaysia, 2011. 377–382
21 Bechikh S, Said L B, Ghédira K. Group preference-based evolutionary multi-objective optimization with non-equally important decision makers: application to the portfolio selection problem. Int J Comput Inf Syst Indus Manag Appl, 2013, 5: 278–288
22 Krasnogor N, Simth J. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evolut Comput, 2005, 9: 474–488
23 Schwefel H P. Evolution and Optimum Seeking. Hoboken: John Wiley & Sons, 1995
24 Nguyen Q H, Ong Y S, Krasnogor N. A study on the design issues of memetic algorithm. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 2007. 2390–2397
25 Le M N, Ong Y S, Jin Y, et al. Lamarckian memetic algorithms: local optimum and connectivity structure analysis. Memetic Comput, 2009, 1: 175–190
26 Sudholt D. The impact of parametrization in memetic evolutionary algorithms. Theor Comput Sci, 2009, 410: 2511–2528
27 Tang J, Lim M H, Ong Y S. Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. Soft Comput, 2007, 11: 873–888
28 Liu D, Tan K C, Goh C K, et al. A multiobjective memetic algorithm based on particle swarm optimization. IEEE Trans Syst Man Cybernet Part B: Cybernet, 2007, 37: 42–50
29 Caponio A, Neri F, Tirronen V. Super-fit control adaptation in memetic differential evolution frameworks. Soft Comput, 2009, 13: 811–831
30 Gong M G, Jiao L C, Liu F, et al. Memetic computation based on regulation between neural and immune systems: the framework and a case study. Sci China Inf Sci, 2010, 53: 1519–1527
31 Smith J E. Coevolving memetic algorithms: a review and progress report. IEEE Trans Syst Man Cybernet Part B: Cybernet, 2007, 37: 6–17
32 Lacca G, Neri F, Mininno E, et al. Ockham's razor in memetic computing: three stage optimal memetic exploration. Inf Sci, 2012, 188: 17–43
33 Meuth R, Lim M H, Ong Y S, et al. A proposition on memes and meta-memes in computing for higher-order learning. Memetic Comput, 2009, 1: 85–100
34 Hadka D, Reed P. Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. Evolut Comput, 2012, 20: 423–452
35 Hadka D, Reed P. Borg: an auto-adaptive many-objective evolutionary computing framework. Evolut Comput, 2013, 21: 231–259
36 Grobler J, Engelbrecht A P, Kendall G, et al. Alternative hyper-heuristic strategies for multi-method global optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, Barcelona, 2010. 1–8
37 Peng F, Tang K, Chen G, et al. Population-based algorithm portfolios for numerical optimization. IEEE Trans Evolut Comput, 2010, 14: 782–800
38 Gong W, Cai Z, Ling C X, et al. Enhanced differential evolution with adaptive strategies for numerical optimization. IEEE Trans Syst Man Cybernet Part B: Cybernet, 2011, 41: 397–413
39 Elsayed S M, Sarker R A, Essam D L. An improved self-adaptive differential evolution algorithm for optimization problems. IEEE Trans Ind Inf, 2013, 9: 89–99
40 Zhang X, Srinivasan R, Liew M V. On the use of multi-algorithm, genetically adaptive multi-objective method for multi-site calibration of the SWAT model. Hydrol Process, 2010, 24: 955–969
41 Dane J H, Vrugt J A, Unsal E. Soil hydraulic functions determined from measurements of air permeability, capillary modeling, and high-dimensional parameter estimation. Vadose Zone J, 2011, 10: 459–465
42 Burke E K, Kendall G, Soubeiga E. A tabu-search hyperheuristic for timetabling and rostering. J Heuristics, 2003, 9: 451–470
43 Beckers M L M, Derks E P P A, Melssen W J, et al. Using genetic algorithms for conformational analysis of biomacromolecules. Comput Chem, 1996, 20: 449–457
44 Fukuyama Y, Chiang H D. A parallel genetic algorithm for generation expansion planning. IEEE Trans Power Syst, 1996, 11: 955–961
45 Tao F, Laili Y J, Xu L, et al. FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans Ind Inf, 2013, 9, 2023–2033
46 Matsumura T, Nakamura M, Okech J, et al. A parallel and distributed genetic algorithm on loosely-coupled multiprocessor systems. IEICE Trans Fund Electr Commun Comput Sci, 1998, 81: 540–546
47 Lourenco H R, Martin O C, Stutzle T. Iterated local search. In: Handbook of Metaheuristics. Beilin: Springer, 2003. 320–353
48 Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony algorithm. J Global Optimiz, 2007, 39: 459–471
49 Geem Z W, Kim J H, Loganathan G V. A new heuristic optimization algorithm: harmony search. Simulation, 2001,

76: 60–68
50 Yang X S, Deb S. Cuckoo search via levy flights. In: IEEE World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), Coimbatore, 2009. 210–214
51 Mladenovic N, Hansen P. Variable neighborhood search. Comput Oper Res, 1997, 24: 1097–1100
52 Feo T A, Resende M G. Greedy randomized adaptive search procedures. J Global Optim, 1995, 6: 109–133
53 Socha K, Dorigo M. Ant colony optimization for continuous domains. Eur J Oper Res, 2008, 185: 1155–1173
54 Hu M, Wu T, Weir J D. An adaptive particle swarm optimization with multiple adaptive methods. IEEE Trans Evolut Comput, 2013, 17: 705–720
55 Suganthan P N, Hansen N, Liang J J, et al. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. KanGAL Report 2005005. 2005
56 Wineberg M, Christensen S. An introduction to statistical analysis for evolutionary computation. In: Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation. New York: ACM, 2008. 2639–2664
57 Tao F, Feng Y, Zhang L, et al. CLPS-GA: a case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Appl Soft Comput, 2014, 19: 264–279
58 Tao F, Cheng Y, Xu L, et al. CCIoT-CMfg: cloud computing and Internet of things based cloud manufacturing service system. IEEE Trans Ind Inf, 2014, 10: 1435–1442
59 Tao F, Zuo Y, Xu L, et al. IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing. IEEE Trans Ind Inf, 2014, 10: 1547–1557
60 Tao F, Zuo Y, Xu L, et al. Internet of things and BOM based life cycle assessment of energy-saving and emission-reduction of product. IEEE Trans Ind Inf, 2014, 10: 1252–1264