2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Self-adaptive Differential Evolution with Local Search Chains for Real-Parameter Single-Objective Optimization

A. K. Qin*†, Ke Tang‡, Hong Pan†, Siyu Xia†
*School of Computer Science and Information Technology, RMIT University, Melbourne, Victoria, Australia
†School of Automation, Southeast University, Nanjing, China
‡School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
Email: kai.qin@rmit.edu.au, ketang@ustc.edu.cn, mspanhong@hotmail.com, xia081@gmail.com

***Abstract*—Differential evolution (DE), as a very powerful population-based stochastic optimizer, is one of the most active research topics in the field of evolutionary computation. Self-adaptive differential evolution (SaDE) is a well- known DE variant, which aims to relieve the practical difficulty faced by DE in selecting among many candidates the most effective search strategy and its associated parameters. SaDE operates with multiple candidate strategies and gradually adapts the employed strategy and its accompanying parameter setting via learning the preceding behavior of already applied strategies and their associated parameter settings. Although highly effective, SaDE concentrates more on exploration than exploitation. To enhance SaDE's exploitation capability while maintaining its exploration power, we incorporate local search chains into SaDE following two different paradigms (Lamarckian and Baldwinian) that differ in the ways of utilizing local search results in SaDE. Our experiments are conducted on the CEC-2014 real-parameter single-objective optimization testbed. The statistical comparison results demonstrate that SaDE with Baldwinian local search chains, armed with suitable parameter settings, can significantly outperform original SaDE as well as classic DE at any tested problem dimensionality.**

I. Introduction

Differential evolution (DE) [1]–[3] is one of the most effective techniques in the field of evolutionary computation for solving real-parameter black-box optimization problems. Since invented by Storn and Price in 1995, DE has attracted considerable attentions of both academic and industrial researchers. Today has already seen a large volume of research works on DE, including algorithmic improvements [4]–[13], theoretical studies [14] and real-world applications [3]. DE features many search strategies, which may significantly influence its optimization performance. It is highly problem-dependent to choose the most suitable strategy and its associated control parameters for DE. Meanwhile, even a chosen search strategy equipped with the best-calibrated parameter setting may not perform consistently well at different search stages. Therefore, many research efforts in DE have been devoted to the adaption of search strategies and parameters. Self-adaptive differential evolution (SaDE) proposed in [9], [10] is a well-known representative among these works.

SaDE avoids the time-consuming task of choosing among many candidates the most effective search strategy and its associated parameters for DE. It operates with multiple candidate strategies that better suit problems of different properties. During the search, SaDE adapts the employed strategy and its accompanying parameter setting via learning the preceding behavior of already applied strategies and their associated parameter settings. In other words, the more effectively a search strategy and its parameters had performed in the past, the more probably they will be chosen to apply for the future. Due to applying the trial-and-error scheme to adapt strategies and parameters, SaDE focuses more on exploration than exploitation during the search. Consequently, some already visited promising regions of the search space may not be well exploited to take advantage of more valuable information therein. To address this issue, we introduce the idea of local search chains proposed by [15] into SaDE, following the Lamarckian and Baldwinian paradigms commonly used in Memetic algorithms (MAs) [16], [17], a.k.a. the synergy of global search and local search. In thus developed two SaDE variants, SaDE$_{LAM}$ incorporates local search results into SaDE's search process while SaDE$_{BAL}$ does not. Both variants have two other parameters in additional to SaDE's two parameters: population size (NP) and learning period (LP), i.e., the local search strength (I_{str}) and the local search ratio (r_L), which are used in local search chains to control how intensive and how often the local search process is executed.

This paper systematically investigates the performance of SaDE$_{LAM}$, SaDE$_{BAL}$, SaDE and classic DE (DE/rand/1/bin) on the recently proposed CEC-2014 real-parameter single-objective optimization testbed consisting of 30 test functions at different problem dimension sizes (10D, 30D and 50D). We employ the Wilcoxon's signed rank test [18] to conduct the pairwise comparison between SaDE and each of its two variants (SaDE$_{LAM}$ and SaDE$_{BAL}$) using two different parameter settings ((I_{str}, r_L): (100, 0.2) and (400, 0.8)). This statistical comparison demonstrates that two SaDE variants significantly outperform SaDE at some problem dimension sizes while performing similar to SaDE at all the other problem dimension sizes. Moreover, SaDE$_{BAL}$ using (I_{str}, r_L): (100, 0.2) significantly outperforms SaDE at any tested problem dimensionality. We also compare two SaDE variants using two different parameter settings, SaDE and DE/rand/1/bin using an advanced statistical hypothesis test (the Iman and Davenport test followed by the Hochberg procedure [19], [20]). This statistical comparison further reveals that SaDE$_{BAL}$ using either of the two examined parameter settings demonstrates the statistically superior performance at any tested problem dimen-

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

sionality. Furthermore, we report in detail the performance of SaDE$_{BAL}$ with (I_{str}, r_L): (100, 0.2) on the CEC-2014 testbed.

The remaining paper is organized as follows. First, SaDE is briefly reviewed in Section II, followed by the introduction of the proposed SaDE with local search chains algorithms in Section III. Section IV reports and analyzes experimental results. Section V concludes this paper and mentions some related future work.

II. Self-adaptive Differential Evolution (SaDE)

The scope of this paper focuses on real-parameter single-objective black-box optimization, aiming to find the optimal values of real-valued decision variables to minimize one real-valued objective function without knowing function characteristics. It can be formulated as:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{R}^D} f(\mathbf{x}), f(\mathbf{x}) \in \mathcal{R}$$

where $\mathbf{x} = \{x^1, \ldots, x^D\} \in \mathcal{R}^D$ is a decision vector composed of D real-valued decision variables.

DE is a population-based stochastic search algorithm very effective in real-parameter black-box optimization. In the context of DE, let $\mathbf{x}_{i,g} = \{x^1_{i,g}, \ldots, x^D_{i,g}\}$ represent the *i*-th decision vector at the *g*-th generation and $\mathbf{P}^g = \{\mathbf{x}_{1,g}, \ldots, \mathbf{x}_{NP,g}\}$ denote the population consisting of *NP* decision vectors at the *g*-th generation. DE firstly creates an initial population $\mathbf{P}^0$ by sampling each decision variable from a uniform distribution between certain prescribed bounds of these variables. Then, at a subsequent generation (e.g., *g*-th generation), a trial vector $\mathbf{u}_{i,g}$ is produced with respect to each decision vector $\mathbf{x}_{i,g}$ (so-called target vector) in the current population using mutation and recombination operations. After that, DE uses the greedy replacement to produce the population at the (*g*+1)-th generation, i.e., $\mathbf{P}^{g+1}$. The above procedure is repeated generation by generation until certain termination criteria are met. Readers can refer to [1], [2] for more details about DE's implementation.

DE features its unique differential mutation scheme and its capability of exploiting contour matching. It has many available trial vector generation strategies, typically denoted by "DE/x/y/z" where x, y and z stand for the base vector generation scheme, the number of population member pairs used to form the vector difference and the recombination scheme, respectively. Typically, DE has three key control parameters: population size (NP), crossover rate (CR) and mutation scale factor (F). The performance of DE crucially depends on the employed search strategy and its associated parameter setting which are highly problem-dependent. On the one hand, using the trial-and-error scheme to select the most suitable strategy and parameter setting may expend prohibitive computational costs. On the other hand, a single strategy even armed with well-calibrated parameters cannot guarantee consistent effectiveness at different search stages since regions of the search space explored at different search stages may not always favor this strategy. Therefore, many recent research efforts have been devoted to the adaptation of strategies and parameters. Among these efforts, a differential evolution with strategy adaptation algorithm named SaDE [9], [10] is a well-known representative.

SaDE avoids the time-consuming strategy and parameter setting selection task. It features a pool of potentially effective yet complementary trial vector generation strategies. During the population's evolution, with respect to each target vector in the population at the current generation, one strategy will be selected from this pool according to strategy selection probabilities, which are computed according to the success rate of each strategy for generating promising trial vectors (those that can enter the population for the next generation) within the number of LP (learning period) preceding generations. This selected strategy is then applied to the corresponding target vector to generate the trial vector. For the parameter setting associated with this selected strategy, SaDE adapts CR while randomizing F. Specifically, it archives the CR values associated with each strategy which had generated promising trial vectors within the preceding LP generations. The median of those recorded CR values with respect to each strategy is computed at the end of the current generation, and used as the mean value of the normal distribution with standard deviation 0.1 to generate the CR values to be used by the corresponding strategy in the next generation. The value of F is randomly sampled from the normal distribution with mean value 0.5 and standard deviation 0.3 to maintain both exploration (large F values) and exploitation (small F values) in the entire course of the search. SaDE leaves NP as a manually specified parameter to be determined based on the available problem knowledge and computational budget.

In SaDE, the initial LP generations accumulate the search behavior to be learnt. During this period, all strategy selection probabilities are set to be equal and the mean value of the normal distribution for generating the CR values is set to 0.5. To avoid invalid selection probabilities when the success rates of all strategies are zero or when a strategy is never chosen within the preceding LP generations, a small constant value (0.01) is introduced as illustrated in Algorithm 1. More details about SaDE's implementation can be found in [9], [10], [21].

III. SaDE with Local Search Chains

Intrinsically, SaDE employs the trail-and-error scheme to gradually adapt the employed trail vector generation scheme and its associated parameter setting. As a result, it devotes much more search efforts on exploration than exploitation. Accordingly, some promising regions of the search space, even had already been identified (i.e., one or more population members falled within these regions), cannot be well exploited to take advantage of more valuable information therein.

Memetic algorithms (MAs) [16], [17], as an emerging hot area in the field of evolutionary computation, study the synergy of population-based global search and separate local search procedures. Usually, these local search procedures can incorporate various available problem knowledge and thus may significantly enhance the quality of finally obtained solutions. Compared to the pure population-based global search, MAs often demonstrate superior efficacy in solving complex optimization problems. However, the performance of MAs relies on several crucial factors, e.g., the properties of local search methods, the local search strength and frequency, the interaction between the population-based search and the local search, etc. Among existing works to addressing these issues [15]–[17], [22], an approach called local search chains [15]

can adapt the strength of the local search to exploit promising regions identified by the population-based global search. The essence of local search chains is to concatenate multiple low-strength local search procedures to provide a single uninterrupted high-strength local search process. To achieve this, the final solution and configuration achieved by the preceding local search execution will be recorded and then used as the initial solution and configuration for the next local search invocation.

To compensate the insufficient exploitation in SaDE while retaining SaDE's good exploration ability, we propose two SaDE variants which incorporate local search chains into SaDE following two paradigms (Lamarckian and Baldwinian) commonly employed in MAs. These two paradigms differ in the ways of utilizing the solutions obtained by the local search in SaDE. Specifically, SaDE with Lamarckian local search chains (SaDE$_{LAM}$) updates SaDE's population with local search results while SaDE with Baldwinian local search chains (SaDE$_{BAL}$) does not.

Let I_{str} and r_L denote the local search strength (the number of function evaluations assigned to one local search execution) and the local search ratio (the number of function evaluations assigned to the local search divided by the maximal number of function evaluations allowed for the entire algorithm), respectively. Accordingly, the relationship between the global search execution and the local search execution can be determined by $I_{str} \cdot (1 - r_L)/r_L$, which defines the number of function evaluations to be executed by the global search between any two consecutive local search executions. The pseudo-code of SaDE with local search chains is illustrated in Algorithm 1. Three key steps related to local search chains (lines: 4, 19 and 24) are elaborated as follows:

- **Initialization** (Algorithm 2): A local search flag set ($\{LSF_i, i = 1, \ldots, NP\}$) and a local search archive set ($\{LSA_i, i = 1, \ldots, NP\}$) are initialized. Each archive LSA_i contains three elements, i.e., the individual used as the starting point of the local search ($\mathbf{x}_{i,0}$), the objective function value of this individual ($f(\mathbf{x}_{i,0})$), and the local search configuration ($lsparam_{default}$). Each flag LSF_i has three status values, i.e., "1" means the individual in LSA_i is not yet exploited; "2" means the individual in LSA_i was obtained by a previous local search execution and can still be further improved; "0" means the individual in LSA_i was obtained by a previous local search execution and cannot be further improved any more.
- **Updating** (Algorithm 3): If a trial vector $\mathbf{u}_{i,g}$ can enter $\mathbf{P}_{g+1}$, it will update the i^{th} element in the local search flag set and the local search archive set, and may also lead to the expansion of the flag and archive sets if $LSF_i = 0$ or 2.
- **Implementation** (Algorithm 4): The best individual stored in the current local search archive will be selected as the starting point of the local search if its corresponding local search flag is above 0. This allows the local search to exploit either previously unexploited (flag=1) or previously less exploited (flag=2) promising regions of the search space. The local search will start from the selected individual using the configuration stored in its corresponding archive. After executing the local search, the achieved solution and configuration will update the corresponding archive. Furthermore, the local search flag corresponding to the local search archive of the selected individual will be set to 0 (if further improvement cannot be expected, e.g., when the change in either the solution space or the objective space is trivial at the end of the last local search execution) and 2 (otherwise). Finally, in the case of Lamarckian local search chains, the solution obtained by the local search will update SaDE's global best and also its corresponding target vector only if the corresponding archive's index does not exceed *NP*.

Algorithm 2 Local Search Chains: Initialization (Line 4 in Algorithm 1)

1: Initialize a set of local search flags $LSF_i = 1, i = 1, \ldots NP$ with index i corresponding to the i^{th} individual in the population.
2: Initialize a set of local search archives $LSA_i = \{\mathbf{x}_{i,0}, f(\mathbf{x}_{i,0}), lsparam_{default}\}, i = 1, \ldots NP$ with index i corresponding to the i^{th} individual in the population. Here, $lsparam_{default}$ denotes the default parameter setting of the local search method.

Algorithm 3 Local Search Chains: Updating (Line 19 in Algorithm 1)

1: Suppose the current size of the local search flag set and archive set is $m, m \geq NP$.
2: **if** ($LSF_i \neq 1$) **then**
3: $LSF_{m+1} = LSF_i$ and $LSA_{m+1} = LSA_i$.
4: **end if**
5: $LSF_i = 1$ and $LSA_i = \{\mathbf{u}_{i,g}, f(\mathbf{u}_{i,g}), lsparam_{default}\}$.

Among the proposed SaDE with local search chains algorithms, SaDE$_{LAM}$ utilizes the solutions obtained by the local search to update the population of SaDE and thus enhances SaDE's exploitation ability at some expense of its exploration power. SaDE$_{BAL}$ compensates the exploitation power of SaDE via separate local search procedures without influencing SaDE's exploration ability. Note that SaDE$_{BAL}$ is distinct from the two-stage global-followed-by-local search approach. Instead of exploiting the best solution achieved by a complete execution of the global search, SaDE$_{BAL}$ may identify a superior solution by exploiting many intermediate promising solutions obtained in the course of the global search. In our implementation, we choose the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [23] as the local search method.

IV. Experiments

We test classic DE (DE/rand/1/bin), SaDE and two SaDE variants (SaDE$_{LAM}$ and SaDE$_{BAL}$) using two different parameter settings ((I_{str}, r_L): (100, 0.2) and (400, 0.8)) on 30 test functions contained in the recently proposed CEC-2014 testbed at three problem dimension sizes (10D, 30D and 50D).

To find out whether local search chains can help to improve the performance of SaDE, we employ the Wilcoxon's signed rank test [18] to make a pairwise comparison between SaDE and each of SaDE$_{LAM}$ and SaDE$_{BAL}$ using two different parameter settings respectively. Furthermore, we conduct an advanced statistical hypothesis test (the Iman and Davenport test followed by the Hochberg procedure [19], [20]) to compare

Algorithm 4 Local Search Chains: Implementation (Line 24 in Algorithm 1)

1: Find the most promising starting point from the current local search archive set to perform the local search: $s = \arg_i \min\{LSA_i(2), LSF_i > 0\}$ where $LSA_i(2)$ stands for the 2^{nd} element in the local search archive LSA_i.
2: Perform the local search for I_{str} function evaluations, starting from $LSA_s(1)$ using the configuration $LSA_s(3)$.
3: Update the local search archive LSA_s with the final solution obtained by the local search, its corresponding objective function value and the current local search configuration.
4: For **Lamarckian Local Search Chain**:
    if $s \leq NP$ and ($LSA_s(2) \leq f(\mathbf{x}_{s,g+1})$) **then** $\mathbf{x}_{s,g+1} = LSA_s(1)$
    if ($LSA_s(2) \leq f(\mathbf{x}_{g+1}^{gbest})$) **then** $\mathbf{x}_{g+1}^{gbest} = LSA_s(1)$
5: **if** ($LSA_s(1)$ cannot be further improved by the local search, e.g., the last local search improvement is trivial) **then**
6: $LSF_s = 0$
7: **else**
8: $LSF_s = 2$
9: **end if**

the performance of six algorithms, i.e., DE/rand/1/bin, SaDE, SaDE$_{LAM}$ and SaDE$_{BAL}$ using two different parameter settings respectively, aiming at finding the statistically superior algorithms. Since both above statistical comparisons reveal that SaDE$_{BAL}$ with (I_{str}, r_L): (100, 0.2) demonstrates the superior performance over all 30 test functions at any examined problem dimensionity, we report, for SaDE$_{BAL}$ with (I_{str}, r_L): (100, 0.2), all performance measures advocated by the CEC-2014 testbed [24] as well as the success rate and the expected running time to succeed (ERT) [25], [26].

A. Experimental Setup

The CEC-2014 testbed involves 30 numerical test functions grouped into uni-modal functions (f_1-f_3), simple multi-modal functions (f_4-f_{16}), hybrid functions (f_{17}-f_{22}) and composition functions (f_{23}-f_{30}), which extends its predecessor (i.e, the CEC-2013 testbed) by introducing several new features, e.g., additional basic problems, composing a test problem by extracting features dimension-wisely from several problems, the graded level of linkages, rotated trap problems and so on. The detailed description of the CEC-2014 testbed can be found in [24].

For each of 30 test functions at each of three problem dimension sizes (10D, 30D and 50D), we test six algorithms including DE/rand/1/bin using its commonly suggested parameter setting (NP: 50, CR: 0.9 and F: 0.5) [2], [27], SaDE using its commonly suggested parameter setting (NP: 50 and LP: 50) [10], [21], SaDE$_{LAM}$ with (I_{str}, r_L): (100, 0.2), so-called SaDE$_{LAM1}$, SaDE$_{LAM}$ with (I_{str}, r_L): (400, 0.8), so-called SaDE$_{LAM2}$, SaDE$_{BAL}$ with (I_{str}, r_L): (100, 0.2), so-called SaDE$_{BAL1}$, and SaDE$_{BAL}$ with (I_{str}, r_L): (400, 0.8), so-called SaDE$_{BAL2}$.

Each of the six algorithms under test is executed 51 times with respect to each test function at each problem dimension size. Each of 51 runs uses distinct random seeds. For any individual run, all algorithms share the same random seed.

TABLE IV. COMPUTATIONAL COMPLEXITY MEASURED BY CPU SECONDS ([24]) AT 10D, 30D AND 50D, RESPECTIVELY. T_0 MEASURES THE COMPUTATION TIME OF BASIC OPERATIONS. T_1 MEASURES THE COMPUTATION TIME OF 200000 EVALUATIONS OF TEST FUNCTION f_{18}. $\widehat{T}_2$ MEASURES THE AVERAGE COMPUTATION TIME FOR FIVE ALGORITHM EXECUTIONS ON TEST FUNCTION f_{18} WITH EACH EXECUTION EXPENDING 200000 EVALUATIONS.

DIM	T_0	T_1	$\widehat{T}_2$	$(\widehat{T}_2 - T_1)/T_0$
$D = 10$	0.165	2.019	14.233	74.023
$D = 30$		2.166	19.916	107.576
$D = 50$		2.667	22.491	120.147

Two stopping criteria are used here [24]: (1) the maximum number of function evaluations (maxFEvals), set to 10^4 times the problem dimension size, is reached. (2) The object function error value (FEV), defined as the difference between the objective function value of the best solution found so far and that of the globally optimal solution, is less than or equal to 10^{-8}. In this case, **we set FEV to** 10^{-8} instead of zero as suggested by the CEC-2014 testbed since the latter way may dramatically decrease the average FEV at termination if only a few runs achieve the FEVs less than or equal to 10^{-8}.

We use MATLAB to implement all algorithms. The algorithm execution platform is a Windows PC with the Intel Xeon E5-2630 CPU at 2.3 GHz.

The algorithm's performance is measured by:

- the best, worse, median and mean (standard deviation) of the FEVs achieved when the algorithm terminates over 51 runs;
- the success rate (SR) over 51 runs. An execution run is claimed to succeed once the algorithm achieves the FEV smaller than 10^{-8};
- the expected running time to succeed (ERT) [25], [26]. This performance index estimates the expected number of function evaluations to succeed. It is computed by the total number of function evaluations when the algorithm succeeds or terminates (without succeeding) summed over 51 runs and divided by the total number of successful runs. If all runs fail, this measure becomes invalid;
- the computational complexity measured by CPU-seconds at three problem dimension sizes (10D, 30D and 50D), respectively [24].

B. Result Analysis

Our first experiment employs the Wilcoxon's signed rank test [18] to conduct the pairwise comparison between the mean FEVs achieved by SaDE and those achieved by each of the four algorithms (SaDE$_{LAM1}$, SaDE$_{LAM2}$, SaDE$_{BAL1}$ and SaDE$_{BAL2}$) for 30 functions at different problem dimensionality (10D, 30D, 50D and the combination of all dimension sizes), respectively. The results reported in Table I demonstrate that each of the four tested algorithms significantly outperforms SaDE at some problem dimension sizes while maintaining the similar performance to SaDE at all the others. This observation verifies that the incorporation of local search chains into SaDE can improve SaDE's performance in a statically significant manner. Notably, the SaDE$_{BAL1}$ algorithm, i.e., SaDE$_{BAL}$ with (I_{str}, r_L): (100, 0.2), can significantly outperform SaDE at any tested problem dimensionality.

TABLE I. WILCOXON'S SIGNED RANK TEST RESULTS OF COMPARING **SADE** (NP:50, LP:50) WITH ITS VARIANTS INCORPORATING LOCAL SEARCH CHAINS, I.E., SADE WITH LAMARCKIAN LOCAL SEARCH CHAINS: **SADE$_{\text{LAM1}}$** AND **SADE$_{\text{LAM2}}$** AS WELL AS SADE WITH BALDWINIAN LOCAL SEARCH CHAINS: **SADE$_{\text{BAL1}}$** AND **SADE$_{\text{BAL2}}$** OVER 30 CEC-2014 TEST FUNCTIONS AT PROBLEM DIMENSIONALITY 10D, 30D, 50D AND ALL (COMBINATION OF ALL DIMENSION SIZES), RESPECTIVELY. HERE, THE MEAN FEVS ARE COMPARED. $R-$ AND $R+$ ARE THE INTERMEDIATE RANKING VALUES USED IN THE WILCOXON'S SIGNED RANK TEST. $R+$ BEING LARGER (SMALLER) THAN $R-$ INDICATES **SADE** PERFORMS WORSE (BETTER) THAN ITS COMPETITOR WITH p-VALUE STATISTICALLY MEASURING THE PERFORMANCE DIFFERENCE AT THE SIGNIFICANT LEVEL OF 0.05. THE COLUMN "WIN-FLAG" INDICATES THAT SADE IS BETTER (1), SIMILAR (0) OR WORSE (-1) THAN ITS COMPETITOR.

SaDE vs.	PROBLEMS															
	10D				30D				50D				ALL			
	R^-	R^+	p-value	win-flag	R^-	R^+	p-value	win-flag	R^-	R^+	p-value	win-flag	R^-	R^+	p-value	win-flag
SaDE$_{\text{LAM1}}$	71	280	7.9523e-003	-1	141	294	9.8092e-002	0	209	256	6.2884e-001	0	1295	2360	1.9633e-002	-1
SaDE$_{\text{LAM2}}$	147	204	4.6916e-001	0	148	287	1.3289e-001	0	116	349	1.6566e-002	-1	1196	2459	5.6560e-003	-1
SaDE$_{\text{BAL1}}$	50	275	2.4697e-003	-1	37	369	1.5679e-004	-1	39	396	1.1351e-004	-1	361	3042	5.7534e-010	-1
SaDE$_{\text{BAL2}}$	145	206	4.3855e-001	0	88	347	5.1070e-003	-1	79	386	1.5927e-003	-1	867	2788	2.5686e-005	-1

TABLE II. PERFORMANCE COMPARISON OF SIX ALGORITHMS (**DE**, **SADE**, **SADE$_{\text{LAM1}}$**, **SADE$_{\text{LAM2}}$**, **SADE$_{\text{BAL1}}$** AND **SADE$_{\text{BAL2}}$**) USING THE IMAN AND DAVENPORT TEST WITH THE HOCHBERG POST-HOC PROCEDURE OVER 30 CEC-2014 TEST FUNCTIONS AT PROBLEM DIMENSIONALITY 10D, 30D, 50D AND ALL (COMBINATION OF ALL DIMENSION SIZES), RESPECTIVELY. HERE, THE MEAN FEVS ARE COMPARED. AMONG SIX ALGORITHMS, THOSE LEADING TO THE STATISTICALLY SIGNIFICANTLY BETTER PERFORMANCE (AT THE SIGNIFICANCE LEVEL OF 0.05) OVER OTHERS WITH RESPECT TO 10D, 30D, 50D AND ALL PROBLEMS RESPECTIVELY ARE DENOTED BY $*$. AN EMPTY CELL MEANS THE CORRESPONDING ALGORITHM IS STATISTICALLY SIGNIFICANTLY WORSE THAN SOME OTHER ALGORITHMS FOR SOLVING ALL 30 TEST FUNCTIONS AT A CERTAIN PROBLEM DIMENSION.

PROBLEMS	DE	SaDE	SaDE$_{\text{LAM1}}$	SaDE$_{\text{LAM2}}$	SaDE$_{\text{BAL1}}$	SaDE$_{\text{BAL2}}$
10D		$*$	$*$	$*$	$*$	$*$
30D				$*$	$*$	$*$
50D				$*$	$*$	$*$
ALL					$*$	$*$

Our second experiment aims to find the statistically superior algorithms among six algorithms (DE/rand/1/bin, SaDE, SaDE$_{LAM1}$, SaDE$_{LAM2}$, SaDE$_{BAL1}$ and SaDE$_{BAL2}$) by using an advanced statistical hypothesis test. Specifically, we first employ the Iman and Davenport test [19], [20] to compare the mean FEVs achieved by six algorithms over all 30 test functions to judge whether at least two algorithms have the statistically significantly different performance. If this is true, we then apply the Hochberg post-hoc procedure [19], [20] to further find out the exact statistically superior algorithms. In our work, the significance level is set to 0.05.

The Iman and Davenport test [19], [20] is an improved version of the well-known Friedman's test [18] for detecting among multiple algorithms whether there exists the statistically significant difference between the performance of at least two algorithms. Although it is able to identify the existence of the performance distinction among multiple compared algorithms, it cannot separate out which algorithms are significantly different from the others. To address this issue, some post-hoc procedures should be employed to perform pairwise performance comparisons under the control of the family-wise error rate (FWER) [19], [20]. In this work, the Hochberg procedure is used, as recommended in [19], [20]. To apply the Hochberg procedure, we choose the control method as the algorithm with the lowest ranking value obtained in the Iman and Davenport test. For more detailed information about the Iman and Davenport test and the Hochberg post-hoc procedure, readers can refer to [18]–[20].

Table II reports the comparison results. We can observe that the effect of incorporating local search chains into SaDE become prominent when solving higher dimensional problems. Furthermore, SaDE with Balwinian local search chains using either of the two tested parameter settings demonstrates the statistically superior performance at any tested problem dimensionality.

Based on the above findings, we report in detail the performance of SaDE$_{BAL1}$, i.e., SaDE$_{BAL}$ with (I_{str}, r_L): (100, 0.2), in Table III. We can observe from this table that SaDE$_{BAL1}$'s performance decreases as increasing the problem dimension size. For 10D problems, SaDE$_{BAL1}$ has non-zero SRs on 10 out of 30 test functions, i.e., f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_{10} and f_{17}. On 13 out of the remaining 20 test functions, i.e., f_9, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}, f_{20}, f_{21} and f_{22}, SaDE$_{BAL1}$ achieves FEVs less than 1.00e+00 in at least one run. For 30D problems, SaDE$_{BAL1}$ has non-zero SRs on six out of 30 test functions, i.e., f_2, f_3, f_4, f_7, f_8 and f_{10}. Among the remaining 24 test functions, SaDE$_{BAL1}$ achieves FEVs less than 1.00e+00 in at least one run on six functions, i.e., f_1, f_6, f_{12}, f_{13} and f_{14}. For 50D problems, SaDE$_{BAL1}$ has non-zero SRs on three out of 30 test functions, i.e., f_4, f_7 and f_8. Among the remaining 27 test functions, SaDE$_{BAL1}$ achieves FEVs less than 1.00e+00 in at least one run on seven functions, i.e., f_1, f_2, f_3, f_{10}, f_{12}, f_{13} and f_{14}. The computational complexity of SaDE$_{BAL1}$ is depicted in Table IV.

V. CONCLUSIONS AND FUTURE WORK

We proposed to incorporate local search chains into the SaDE algorithm following two paradigms (Lamarckian and Baldwinian) commonly used in MAs. The two developed SaDE with local search chain algorithms, i.e., SaDE$_{LAM}$ and SaDE$_{BAL}$), differ in how to utilize local search results in SaDE. In fact, SaDE$_{LAM}$ enhances SaDE's exploitation ability at some expense of its exploration power. SaDE$_{BAL}$ compensates the exploitation power of SaDE via separate local search procedures without influencing SaDE's exploration ability.

We tested classic DE (DE/rand/1/bin), SaDE as well as SaDE$_{LAM}$ and SaDE$_{BAL}$ using two different parameter settings ((I_{str}, r_L): (100, 0.2) and (400, 0.8)) respectively on 30 CEC-2014 test functions at different problem dimen-

TABLE III. PERFORMANCE (PFM) OF **SaDE**$_{BAL}$ WITH THE PARAMETER SETTING NP:50, LP:50, $I_{str} = 100$, $r_L = 0.2$ AT PROBLEM DIMENSIONALITY (DIM) 10D, 30D AND 50D, RESPECTIVELY. **BEST**, **WORST**, **MEDIAN**, **MEAN** (**STD**) REPRESENT THE BEST, WORST, MEDIAN, MEAN (STANDARD DEVIATION) OF THE FEVS AT EXECUTION TERMINATION OVER 51 RUNS, RESPECTIVELY. **SR** AND **ERT** STAND FOR THE SUCCESS RATE AND THE EXPECTED RUNNING TIME TO SUCCEED. **ERT** IS DENOTED BY "-" (INVALID) WHEN ALL 51 RUNS FAIL.

DIM	PFM	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
10D	**Best**	1.00e-08	1.00e-08	1.00e-08	1.00e-08	1.00e-08	1.00e-08	1.00e-08	1.00e-08	9.95e-01	1.00e-08	1.87e-01	2.60e-02	1.94e-02	3.90e-02	2.27e-01
	Worst	7.92e-04	1.00e-08	1.00e-08	1.00e-08	2.00e+01	8.95e-01	1.23e-02	1.00e-08	4.97e+00	1.87e-01	1.34e+02	1.93e-01	6.44e-02	2.00e-01	6.58e-01
	Median	1.00e-08	1.00e-08	1.00e-08	1.00e-08	2.00e+01	1.00e-08	1.00e-08	1.00e-08	2.98e+00	1.00e-08	1.86e+01	9.26e-02	3.19e-02	9.42e-02	3.88e-01
	Mean	2.23e-05	1.00e-08	1.00e-08	1.00e-08	1.70e+01	1.76e-02	2.46e-03	1.00e-08	2.71e+00	1.96e-02	3.41e+01	9.45e-02	3.50e-02	9.62e-02	3.95e-01
	Std	1.20e-04	5.01e-24	5.01e-24	5.01e-24	6.83e+00	1.25e-01	4.12e-03	5.01e-24	9.14e-01	4.05e-02	3.75e+01	4.47e-02	1.11e-02	3.64e-02	9.32e-02
	SR	0.94	1.00	1.00	1.00	0.04	0.92	0.73	1.00	0.00	0.76	0.00	0.00	0.00	0.00	0.00
	ERT	5.09e+04	2.59e+04	2.09e+04	5.24e+03	2.54e+06	5.28e+04	5.84e+04	1.86e+04	-	6.87e+04	-	-	-	-	-
		f16	**f17**	**f18**	**f19**	**f20**	**f21**	**f22**	**f23**	**f24**	**f25**	**f26**	**f27**	**f28**	**f29**	**f30**
	Best	6.81e-01	1.00e-08	3.15e-03	4.98e-02	1.20e-02	1.51e-05	1.51e-03	3.29e+02	1.00e+02	1.00e+02	1.00e+02	1.03e+00	3.14e+02	1.80e+02	2.62e+02
	Worst	2.29e+00	1.78e+02	7.11e+00	4.54e-01	1.11e+00	1.69e+01	3.62e-01	3.29e+02	1.12e+02	2.00e+02	1.00e+02	4.00e+02	4.81e+02	2.37e+02	6.00e+02
	Median	1.44e+00	1.24e+01	1.01e+00	1.69e-01	1.21e-01	1.39e-01	3.16e-02	3.29e+02	1.09e+02	1.20e+02	1.00e+02	2.16e+00	3.60e+02	2.24e+02	4.83e+02
	Mean	1.43e+00	3.35e+01	1.10e+00	1.90e-01	1.66e-01	1.82e+00	5.51e-02	3.29e+02	1.09e+02	1.34e+02	1.00e+02	1.46e+02	3.74e+02	2.23e+02	4.84e+02
	Std	3.65e-01	4.76e+01	1.20e+00	8.01e-02	1.98e-01	4.99e+00	7.41e-02	2.87e-13	2.30e+00	2.97e+01	9.44e-03	1.70e+02	3.58e+01	8.47e+00	5.92e+01
	SR	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	ERT	-	5.04e+06	-	-	-	-	-	-	-	-	-	-	-	-	-

DIM	PFM	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
30D	**Best**	1.09e-03	1.00e-08	1.00e-08	1.00e-08	2.00e+01	5.31e-01	1.00e-08	1.00e-08	1.19e+01	1.00e-08	9.25e+02	9.61e-02	8.82e-02	1.34e-01	1.42e+00
	Worst	3.15e-01	1.00e-08	2.27e-01	1.00e-08	2.00e+01	8.04e+00	3.20e-02	9.95e-01	3.38e+01	1.14e+00	2.11e+03	3.92e-01	2.16e-01	2.72e-01	4.13e+00
	Median	1.47e-03	1.00e-08	1.00e-08	1.00e-08	2.00e+01	3.68e+00	1.00e-08	1.00e-08	2.19e+01	4.16e-02	1.56e+03	1.90e-01	1.53e-01	2.13e-01	2.55e+00
	Mean	1.35e-02	1.00e-08	7.15e-03	1.00e-08	2.00e+01	3.64e+00	2.56e-03	1.95e-02	2.26e+01	6.44e-02	1.58e+03	2.09e-01	1.50e-01	2.10e-01	2.61e+00
	Std	5.75e-02	5.01e-24	3.43e-02	5.01e-24	2.01e-04	1.68e+00	7.13e-03	1.39e-01	5.15e+00	1.56e-01	2.65e+02	8.11e-02	3.10e-02	3.53e-02	6.17e-01
	SR	0.00	1.00	0.86	1.00	0.00	0.00	0.86	0.98	0.00	0.16	0.00	0.00	0.00	0.00	0.00
	ERT	-	8.66e+04	1.60e+05	3.54e+04	-	-	6.51e+04	6.60e+04	-	1.79e+06	-	-	-	-	-
		f16	**f17**	**f18**	**f19**	**f20**	**f21**	**f22**	**f23**	**f24**	**f25**	**f26**	**f27**	**f28**	**f29**	**f30**
	Best	8.56e+00	3.06e+02	4.52e+01	3.27e+00	1.50e+01	1.92e+01	2.06e+01	3.14e+02	2.23e+02	2.00e+02	1.00e+02	3.21e+02	7.38e+02	4.39e+02	6.17e+02
	Worst	1.06e+01	2.58e+03	1.74e+02	8.61e+00	3.55e+02	2.07e+03	3.63e+02	3.14e+02	2.41e+02	2.14e+02	2.00e+02	5.28e+02	9.54e+02	1.80e+03	2.13e+03
	Median	9.72e+00	1.01e+03	8.39e+01	4.63e+00	6.87e+01	7.54e+02	1.41e+02	3.14e+02	2.26e+02	2.09e+02	1.00e+02	4.01e+02	8.21e+02	8.28e+02	1.06e+03
	Mean	9.68e+00	1.08e+03	9.07e+01	4.89e+00	9.28e+01	7.57e+02	1.03e+02	3.14e+02	2.28e+02	2.08e+02	1.04e+02	4.01e+02	8.34e+02	9.25e+02	1.10e+03
	Std	4.49e-01	5.05e+02	3.04e+01	1.12e+00	7.72e+01	3.81e+02	7.15e+01	2.93e-10	4.11e+00	4.60e+00	1.96e+01	3.88e+01	5.01e+01	2.41e+02	3.16e+02
	SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	ERT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

DIM	PFM	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
50D	**Best**	1.03e-02	1.57e-03	5.57e-03	1.00e-08	2.00e+01	7.12e+00	1.00e-08	1.00e-08	3.28e+01	1.25e-02	2.75e+03	1.12e-01	1.76e-01	1.97e-01	5.10e+00
	Worst	1.21e+02	4.91e+01	1.21e+03	3.99e+00	2.00e+01	2.33e+01	3.19e-02	9.95e-01	8.56e+01	1.39e+00	4.56e+03	4.30e-01	4.55e-01	3.46e-01	1.90e+01
	Median	7.57e-01	4.43e-03	7.21e+00	1.00e-08	2.00e+01	1.58e+01	7.40e-03	1.00e-08	6.17e+01	6.25e-02	3.73e+03	2.46e-01	3.02e-01	2.76e-01	1.03e+01
	Mean	1.25e+01	1.19e+00	4.72e+01	3.91e-01	2.00e+01	1.54e+01	6.23e-03	7.80e-02	6.12e+01	1.54e-01	3.70e+03	2.45e-01	2.98e-01	2.71e-01	1.10e+01
	Std	2.32e+01	6.90e+00	1.81e+02	1.20e+00	2.31e-05	3.22e+00	6.95e-03	2.70e-01	1.01e+01	2.75e-01	3.84e+02	7.00e-02	5.90e-02	3.68e-02	3.06e+00
	SR	0.00	0.00	0.00	0.90	0.00	0.00	0.45	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	ERT	-	-	-	1.10e+05	-	-	6.69e+05	1.53e+05	-	-	-	-	-	-	-
		f16	**f17**	**f18**	**f19**	**f20**	**f21**	**f22**	**f23**	**f24**	**f25**	**f26**	**f27**	**f28**	**f29**	**f30**
	Best	1.75e+01	1.28e+03	6.04e+01	9.96e+00	5.32e+01	7.76e+02	1.56e+02	3.37e+02	2.69e+02	2.00e+02	1.00e+02	5.88e+02	1.06e+03	7.12e+02	1.42e+03
	Worst	2.01e+01	8.21e+04	2.86e+02	7.38e+01	7.25e+02	1.14e+05	7.93e+02	3.37e+02	2.89e+02	2.32e+02	2.00e+02	9.08e+02	1.45e+03	1.86e+03	5.95e+03
	Median	1.88e+01	3.21e+03	1.45e+02	3.13e+01	2.19e+02	3.36e+03	4.57e+02	3.37e+02	2.77e+02	2.23e+02	2.00e+02	7.85e+02	1.19e+03	9.99e+02	2.21e+03
	Mean	1.88e+01	9.32e+03	1.48e+02	2.82e+01	2.56e+02	1.10e+04	4.40e+02	3.37e+02	2.77e+02	2.22e+02	1.55e+02	7.65e+02	1.21e+03	1.08e+03	2.29e+03
	Std	6.13e-01	1.34e+04	4.14e+01	1.59e+01	1.68e+02	2.07e+04	1.48e+02	7.82e-09	3.73e+00	7.47e+00	5.01e+01	7.91e+01	9.08e+01	2.25e+02	7.83e+02
	SR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	ERT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

sionality. Both pairwise and multiple statistical comparisons were conducted to verify the effectiveness of the proposed algorithms, which revealed that the incorporation of local search chains into SaDE led to the statistically better or similar performance compared to SaDE at any tested problem dimensionality. Furthermore, the SaDE$_{BAL}$ with (I_{str}, r_L): (100, 0.2) algorithm consistently demonstrated the statistically superior performance at any tested problem dimensionality. Therefore, we comprehensively reported its performance on the CEC-2014 testbed. Our future work includes analyzing the time complexity of the proposed algorithms, developing more efficient local search methods, investigating the effects of two parameters I_{str} and r_L on handling problems of different properties, and studying the performance of the proposed algorithms for solving high-dimensional (large-scale) problems.

ACKNOWLEDGMENT

This work was supported by NSFC under Grant No. 61005051 and SRFDP under Grant No. 20100092120027.

REFERENCES

[1] R. M. Storn and K. V. Price, "Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute, Berkeley, CA, USA, ICSI Technical Report 95-012, 1995.

[2] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*. Springer, 2005.

[3] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[4] H. A. Abbass, "The self-adaptive pareto differential evolution algorithm," in *Proc.of the 2002 IEEE Congress on Evolutionary Computation (CEC'02)*, vol. 1, Honolulu, Hawaii, USA, 2002, pp. 831–836.

[5] J. Liu and J. A. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, Jun. 2004.

[6] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. of the 2005 conference on Genetic and evolutionary computation (GECCO'05)*. ACM, 2005, pp. 991–998.

[7] M. Omran, A. Salman, and A. Engelbrecht, "Self-adaptive differential evolution," *Computational intelligence and security*, pp. 192–199, 2005.

[8] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[9] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. of the 2005 IEEE Congress on Evolutionary Computation (CEC'05)*. IEEE, 2005, pp. 1785–1791.

[10] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[11] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Computing*, vol. 15, no. 11, pp. 2141–2155, Sep. 2010.

[12] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, Feb. 2011.

[13] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, Mar. 2011.

[14] D. Zaharie, "Critical values for the control parameters of differential evolution algorithms," in *Proc. of MENDEL 2002: the 8th International Conference on Soft Computing*, Jun. 2002, pp. 62–67.

[15] D. Molina, M. Lozano, C. Garcia-Martinez, and F. Herrera, "Memetic algorithms for continuous optimisation based on local search chains," *Evolutionary Computation*, vol. 18, no. 1, pp. 27–63, 2010.

[16] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, 2011.

[17] F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms*. Springer, 2012.

[18] D. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC PressI Llc, 2004.

[19] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.

[20] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[21] A. K. Qin, X. Li, H. Pan, and S. Xia, "Investigation of self-adaptive differential evolution on the cec-2013 real-parameter single-objective optimization testbed," in *Proc. of the 2013 IEEE Congress on Evolutionary Computation (CEC'13)*. IEEE, 2013, pp. 1107–1114.

[22] A. György and L. Kocsis, "Efficient multi-start strategies for local search algorithms," *Journal of Artificial Intelligence Research*, vol. 41, no. 2, pp. 407–444, 2011.

[23] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2006.

[24] J. J. Liang, B.-Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201311, 2013.

[25] K. V. Price, "Differential evolution vs. the functions of the 2^{nd} ICEO," in *Proc. of the 1997 IEEE Congress on Evolutionary Computation (CEC'97)*. IEEE, 1997, pp. 153–157.

[26] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking: Experimental setup," INRIA, France, Technical Report, 2013.

[27] A. K. Qin and X. Li, "Differential evolution on the cec-2013 single-objective continuous optimization testbed," in *Proc. of the 2013 IEEE Congress on Evolutionary Computation (CEC'13)*. IEEE, 2013, pp. 1099–1106.

Algorithm 1 SaDE with Local Search Chains

Input: *NP*, *LP*, I_{str}, r_L

Initialize the generation counter $g = 0$, strategy selection probabilities $stPb_{k,g} = 1/4, k = 1, \ldots, 4$, and $CRm_{k,g} = 0.5, k = 1, 2, 3$; Set the success and failure archives to empty.
Initialize the population $\mathbf{P}_g$ of *NP* D-dimensional individuals: $\mathbf{P}_g = \{\mathbf{x}_{1,g}, \ldots, \mathbf{x}_{NP,g}\}$ with $\mathbf{x}_{i,g} = \{x^1_{i,g}, \ldots, x^D_{i,g}\}$.
Evaluate the objective function value of each individual in $\mathbf{P}_g$, i.e., $f(\mathbf{x}_{i,g}), i = 1, \ldots, NP$; Find $\mathbf{x}_g^{gbest} = \mathbf{x}_{i^*,g}$ with $i^* = \arg_i \min f(\mathbf{x}_{i,g})$; Set the evaluation counter $\#feval = NP$.
Initialize the local search flag set ($LSF_i, i = 1, \ldots, NP$) and archive set ($LSA_i, i = 1, \ldots, NP$) (Algorithm 2).
while the predefined termination criteria are not met **do**
  for $i = 1 \rightarrow NP$ **do**
    Select a strategy index k_i in $\{1, 2, 3, 4\}$ based on $stPb_{k,g}, k = 1, \ldots, 4$ using stochastic universal sampling.
    Randomly generate a F value according to normal distribution $rand_n(0.5, 0.3)$.
    Randomly select in $\{1, \ldots, NP\}$ five mutually exclusive indices $r_m, m = 1, \ldots, 5$ that are distinct from i.
    Generate a mutant vector $\mathbf{v}_{i,g} = \{v^1_{i,g}, \ldots, v^D_{i,g}\}$:
      if (k_i == 1) **then** $\mathbf{v}_{i,g} = \mathbf{x}_{r_1,g} + F \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g})$
      if (k_i == 2) **then** $\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_g^{gbest} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g})$
      if (k_i == 3) **then** $\mathbf{v}_{i,g} = \mathbf{x}_{r_1,g} + F \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}) + F \cdot (\mathbf{x}_{r_4,g} - \mathbf{x}_{r_5,g})$
      if (k_i == 4) **then** $\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + rand_u(0, 1) \cdot (\mathbf{x}_{r_1,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g})$
    Generate a trial vector $\mathbf{u}_{i,g} = \{u^1_{i,g}, \ldots, u^D_{i,g}\}$:
      if ($k_i < 4$) **then**
        Randomly generate $CR \in [0, 1]$ from normal distribution $rand_n(CRm_{k_i}, 0.1)$; $j_{rand} = ceil(rand_u(1, D))$.
        for $j = 1 \rightarrow D$ **do**
          $$u^j_{i,g} = \begin{cases} v^j_{i,g} & \text{if } rand_u(0,1) \le CR \text{ or } j = j_{rand} \\ x^j_{i,g} & \text{otherwise} \end{cases}$$
        end for
      else
        $\mathbf{u}_{i,g} = \mathbf{v}_{i,g}$
      end if
  end for
  Set $\mathbf{x}_{i,g+1} = \mathbf{x}_{i,g}, i = 1, \ldots, NP$ and $\mathbf{x}_{g+1}^{gbest} = \mathbf{x}_g^{gbest}$.
  for $i = 1 \rightarrow NP$ **do**
    Function evaluation of the generated trial vector $\mathbf{u}_{i,g}$; Increase the evaluation counter: $\#feval = \#feval + 1$.
    if ($f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g+1})$) **then**
      $\mathbf{x}_{i,g+1} = \mathbf{u}_{i,g}$, and store the tuple (g, k_i, CR) (if k_i == 1, 2 or 3) or (g, k_i) (if k_i == 4) into the success archive.
      if ($f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{g+1}^{gbest})$) **then** $\mathbf{x}_{g+1}^{gbest} = \mathbf{u}_{i,g}$.
      Update the local search flag set and the local search archive set (Algorithm 3).
    else
      Store the tuple (g, k_i) into the failure archive.
    end if
    if ($rem(\#feval - NP, I_{str}/r_L) + 1 == I_{str} \cdot (1 - r_L)/r_L$) **then**
      Perform the local search for I_{str} function evaluations and update the current population (Algorithm 4).
      Increase the evaluation counter: $\#feval = \#feval + I_{str}$.
    end if
  end for
  if ($g \ge LP$) **then**
    if ($g > LP$) **then**
      Remove those tuples with the first elements smaller or equal to $g - LP$ from the success and failure archives.
    end if
    Calculate $S_{k,g}$ and $F_{k,g}$, $k = 1, \ldots, 4$ as the number of tuples having the second elements equal to k in the success and failure archives, respectively.
    if ($S_{k,g} + F_{k,g} > 0$) **then**
      $stPb_{k,g} = S_{k,g}/(S_{k,g} + F_{k,g}) + 0.01$
    else
      $stPb_{k,g} = 0.01$
    end if
    $stPb_{k,g} = stPb_{k,g} / \sum_{k=1,\ldots,4} stPb_{k,g}$
    Calculate $CRm_{k,g}, k = 1, 2, 3$ as the median value of the third elements in those tuples in the success archive having the second elements equal to $k, k = 1, 2, 3$.
  end if
  Increase the generation counter: $g = g + 1$.
end while
Find the optimal solution $\mathbf{x}^*$ via comparing $\mathbf{x}_g^{best}$ and all solutions stored in the local search archive.

NOTE: (1) $rand_u(a, b)$ is uniform sampling in $[a, b]$; (2) $rand_n(a, b)$ is Gaussian sampling with mean a and standard deviation b; (3) $ceil(c)$ is the smallest integer not below c; (4) $rem(a, b)$ is the reminder of a divided by b.