Improving Differential Evolution With a Successful-Parent-Selecting Framework

Shu-Mei Guo, *Member, IEEE*, Chin-Chang Yang, *Student Member, IEEE*, Pang-Han Hsu, *Student Member, IEEE*, and Jason S.-H. Tsai

***Abstract*—An effective and efficient successful-parent-selecting framework is proposed to improve the performance of differential evolution (DE) by providing an alternative for the selection of parents during mutation and crossover. The proposed method adapts the selection of parents by storing successful solutions into an archive, and the parents are selected from the archive when a solution is continuously not updated for an unacceptable amount of time. The proposed framework provides more promising solutions to guide the evolution and effectively helps DE escaping the situation of stagnation. The simulation results show that the proposed framework significantly improves the performance of two original DEs and six state-of-the-art algorithms in four real-world optimization problems and 30 benchmark functions.**

***Index Terms*—Differential evolution (DE), global numerical optimization, parent adaptation, stagnation.**

I. Introduction

DIFFERENTIAL evolution (DE), introduced by Storn and Price [1], [2] and Storn *et al*. [3], is a simple yet efficient evolutionary algorithm (EA) for global optimization problems in continuous search domain. DE and its improved variants have been achieving top ranks in many competitions held under the IEEE Congress on Evolutionary Computation (CEC) conference series [4]–[6]. In the last decade, DE has been widely used in diverse domains of real-world optimization problems, such as electrical engineering [7]–[9], image processing [10]–[12], and graph theory [13]. A detailed survey on the state-of-the-art research for DE can be seen in [14].

Generally, DE begins by initializing a population of candidate solutions. The quality of each solution is evaluated by a fitness function, which represents the single-objective optimization problem. DE selects a set of solutions as parents and evolves the parents through mutation and crossover, which generate a child. A selection process is applied to compare each pair of a predefined parent and its corresponding child in terms of the fitness function, and then the promising one is selected to survive to the next iteration. A child that survives to the next iteration is called a successful solution. DE repeats this procedure until a predefined termination criterion is reached. The best solution found by this procedure is expected to be a near-optimal solution for the optimization problem.

DE, however, may gradually stop generating successful solutions even though the population has not converged to a fixed point. This situation is commonly referred to as stagnation. To steadily generate successful solutions, many advances have been made in the area of self-adaptive control parameters, such as jDE [57], DE with global and local neighborhoods (DEGL) [15], self-adaptive differential evolution (SaDE) [16], JADE [17], ensemble of mutation strategies and parameters with differential evolution (EPSDE) [18], MDE_pBX [19], and success-history adaptation differential evolution (SHADE) [20], [21]. In these methods, the control parameter that was used to generate successful solutions is propagated to the following iterations. The basic operation is thus to record recent control parameters and use them to guide the generation of new control parameter values. SaDE [16] extends the parameter adaptation to strategy adaptation. In SaDE, four strategies are stored in a strategy pool. The strategy that generated successful solutions will gain more probability to be used in the following iterations. In this way, a more suitable strategy can be determined adaptively. Except the parameter adaptation and strategy adaptation, the selection of parents is also a potential aspect to adapt. The proximity-based mutation operators [22] propose an alternative to the uniform random selection of parents during mutation but the selection of parents, which can generate successful solutions, is not utilized.

In this paper, we propose an alternative to the selection of parents during mutation and crossover. The proposed framework adapts the selection of parents by storing successful solutions into an archive, and selects parents from the archive when stagnation has occurred. The proposed framework can be applied to any DE algorithms. We incorporate this scheme to two classical DE algorithms and six state-of-the-art DE variants. Their performance gains are observed over four CEC 2011 [23] real-world optimization problems and 30 benchmark functions in CEC 2014 [24].

The remainder of this paper is organized as follows. Section II briefly introduces DE algorithm. Section III presents the related works of the proposed method. Section IV shows the stagnation of DE and provides the motivation of the

Manuscript received June 13, 2014; revised August 29, 2014; accepted November 21, 2014. Date of publication December 2, 2014; date of current version September 29, 2015. This work was supported by the National Science Council of Taiwan under Grant NSC 102-2221-E-006-199.

S.-M. Guo, C.-C. Yang, and P.-H. Hsu are with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan (e-mail: guosm@mail.ncku.edu.tw).

J. S.-H. Tsai is with the Control System Laboratory, Department of Electrical Engineering, National Cheng Kung University, Tainan 701, Taiwan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2014.2375933

proposed successful-parent-selecting (SPS) framework, which is shown in Section V. In Section VI, we present a comparison of DE algorithms with and without the proposed framework. Finally, this paper concludes in Section VII.

II. DE

DE is a population-based stochastic search method which minimizes an objective function by evolving a population of NP D-dimensional parameter vectors

$$\mathbf{P}_G = \left\{\vec{x}_{i,G} \,|\vec{x}_{i,G} = \left[x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}\right]^T \quad i = 1,\ 2, \ldots, \mathrm{NP}\right\} \tag{1}$$

where G is the generation number. The initial population $\mathbf{P}_0$ covers the entire search space by uniformly distributed random variables between the lower and upper bounds

$$\vec{x}_{\min} = \left[x_{1,\min}, x_{2,\min},\ \ldots\ , x_{D,\min}\right]^T \tag{2}$$

$$\vec{x}_{\max} = \left[x_{1,\max}, x_{2,\max},\ \ldots\ , x_{D,\max}\right]^T. \tag{3}$$

For each generation, certain mutation and crossover operations are employed to produce trial vectors in the current population. Then, a selection operation determines which vectors will survive to the next generation.

A. Mutation

Mutation is a change or perturbation with a random element on the population to locate better solutions. DE creates a mutant vector $\vec{v}_{i,G}$, which is also called donor vector, by utilizing one or more difference vectors. Generally, there are two types of mutation operators in DE families.

1) Type-1

$$\begin{aligned}\vec{v}_{i,G} &= \mathrm{mutate}_1\left(\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}, F\right)\\ &= \vec{p}_{1,i,G} + F \cdot \left(\vec{p}_{2,i,G} - \vec{p}_{3,i,G}\right).\end{aligned} \tag{4}$$

2) Type-2

$$\begin{aligned}\vec{v}_{i,G} &= \mathrm{mutate}_2\left(\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}, \vec{p}_{4,i,G}, \vec{p}_{5,i,G}, F_1, F_2\right)\\ &= \vec{p}_{1,i,G} + F_1 \cdot \left(\vec{p}_{2,i,G} - \vec{p}_{3,i,G}\right) + F_2 \cdot \left(\vec{p}_{4,i,G} - \vec{p}_{5,i,G}\right)\end{aligned} \tag{5}$$

where F, F_1, and F_2 are scaling factors of the difference vectors, and $\{\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}, \vec{p}_{4,i,G}, \vec{p}_{5,i,G}\} \subset \mathbf{P}_G$ are the parents of the ith donor vector in the Gth generation.

There are diverse methods for the selection of parents in DE families. The following are the five most frequently used mutation strategies in the literature.

1) "DE/rand/1"

$$\vec{v}_{i,G} = \mathrm{mutate}_1\left(\vec{x}_{r_1,G}, \vec{x}_{r_2,G}, \vec{x}_{r_3,G}, F\right). \tag{6}$$

2) "DE/best/1"

$$\vec{v}_{i,G} = \mathrm{mutate}_1\left(\vec{x}_{\mathrm{best},G}, \vec{x}_{r_1,G}, \vec{x}_{r_2,G}, F\right). \tag{7}$$

3) "DE/current-to-best/1"

$$\vec{v}_{i,G} = \mathrm{mutate}_2\left(\vec{x}_{i,G}, \vec{x}_{\mathrm{best},G}, \vec{x}_{i,G}, \vec{x}_{r_1,G}, \vec{x}_{r_2,G}, F_1, F_2\right). \tag{8}$$

4) "DE/rand/2"

$$\vec{v}_{i,G} = \mathrm{mutate}_2\left(\vec{x}_{r_1,G}, \vec{x}_{r_2,G}, \vec{x}_{r_3,G}, \vec{x}_{r_4,G}, \vec{x}_{r_5,G}, F_1, F_2\right). \tag{9}$$

5) "DE/best/2"

$$\vec{v}_{i,G} = \mathrm{mutate}_2\left(\vec{x}_{\mathrm{best},G}, \vec{x}_{r_1,G}, \vec{x}_{r_2,G}, \vec{x}_{r_3,G}, \vec{x}_{r_4,G}, F_1, F_2\right) \tag{10}$$

where the indices r_1, r_2, r_3, r_4, and r_5 are mutually exclusive integers randomly selected from the set $I = \{1, 2, \ldots, \mathrm{NP}\}$, and all are different from the index i. The best vector $\vec{x}_{\mathrm{best},G}$ is of the best fitness in the population at generation G. In (8)–(10), the scaling factors $F_1 = F_2$ are of the same values.

B. Crossover

After mutation, DE generates a trial vector $\vec{u}_{i,G}$ by crossover operation to enhance the potential diversity of the population. In the basic version, the binomial crossover operation of DE can be generalized as

$$\begin{aligned}\vec{u}_{i,G} &= \mathrm{xover}\left(\vec{p}_{6,i,G}, \vec{v}_{i,G}, \mathrm{CR}\right)\\ &= \vec{p}_{6,i,G} + \vec{e}_{i,G} \circ \left(\vec{v}_{i,G} - \vec{p}_{6,i,G}\right), \quad \text{for } i = 1, 2, \ldots, \mathrm{NP}\end{aligned} \tag{11}$$

where $\vec{p}_{6,i,G} \in \mathbf{P}_G$ is the parent of the trial vector, and $\vec{e}_{i,G} = [e_{1,i,G}, \ldots, e_{D,i,G}]^T$ is a mask containing a random number of 0 and 1 according to

$$e_{j,i,G} = \begin{cases} 1, & \text{if } \left(\mathrm{rand}_{j,i,G}[0, 1] \le \mathrm{CR}\right) \text{or } (j = j_{\mathrm{rand}}) \\ 0, & \text{otherwise.}\end{cases} \tag{12}$$

The notation $\mathrm{rand}_{j,i,G}[0, 1]$ represents a uniformly distributed random variable within the range [0, 1]. The crossover rate (CR) is a user-specified real value between 0 and 1 to control the fraction of variables copied from the donor vector. j_{rand} is an integer randomly chosen from 1 to D, which ensures that the trial vector gets at least one component from the donor vector. The operator between $\vec{e}_{i,G}$ and $(\vec{v}_{i,G} - \vec{p}_{6,i,G})$ in (11) is the Hadamard product, which is also known as the element-wise product. A similar definition of the generalized crossover operator can be seen in [25].

C. Selection

Selection compares the target vector $\vec{x}_{i,G}$ with the trial vector $\vec{u}_{i,G}$ in terms of the objective function value to determine which vector may survive to the next generation

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G}, & \text{if } f\left(\vec{u}_{i,G}\right) \le f\left(\vec{x}_{i,G}\right) \\ \vec{x}_{i,G}, & \text{otherwise.}\end{cases} \tag{13}$$

The operation in (13) is called a successful update if the trial vector survives to the next generation. The corresponding solution and control parameters are called successful solution and successful control parameters, respectively. Otherwise, they are called unsuccessful update, unsuccessful solution, and unsuccessful control parameters.

To present the overall procedure of DE, we show DE/rand/1 algorithm with the binomial crossover operation in Table I.

TABLE I
PSEUDOCODE OF THE OVERALL DE/RAND/1 ALGORITHM WITH THE BINOMIAL CROSSOVER OPERATION

Algorithm 1 DE/rand/1 algorithm with the binomial crossover operation

Input: f: Objective function.
$MaxGen$: Maximal generation number.
$\vec{x}_{\min}$: Lower bounds.
$\vec{x}_{\max}$: Upper bounds.
NP: Population size.
F: Scaling factor.
CR: Crossover rate.

Initialize a population $\mathbf{P}_0 = \{\vec{x}_{1,0}, \ldots, \vec{x}_{NP,0}\}$ within the lower bounds $\vec{x}_{\min}$ and upper bounds $\vec{x}_{\max}$.
for $G = 0$ to $MaxGen$ **do**
  for $i = 1$ to NP **do**
    Randomly select $r_1 \neq r_2 \neq r_3$ from the set $I = \{1, 2, \ldots, NP\}$.
    Let $\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}$, and $\vec{p}_{4,i,G}$ be the r_1th, r_2th, r_3th, and ith vectors of $\mathbf{P}_G$, respectively.
    Generate a donor vector $\vec{v}_{i,G} = \text{mutate}_1(\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}, F)$.
    Generate a trial vector $\vec{u}_{i,G} = \text{xover}(\vec{p}_{4,i,G}, \vec{v}_{i,G}, CR)$.
    if $f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G})$
      $\vec{x}_{i,G+1} = \vec{u}_{i,G}$;
    else
      $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
    end if
  end for
end for

Output: the candidate solution with the smallest objective function value.

III. RELATED WORKS

This section briefly reviews some recently improved DE variants which are related to the proposed method in this paper. Sections III-A and III-B show recent improvement on mutation strategies and crossover operators, respectively. Section III-C reviews some novel methods on pooling multiple mutation strategies. Section III-D surveys recent advancement of adaptive DE algorithms on strategy adaptation. Section III-E introduces recent DEs on control parameter adaptation. Section III-F discovers recent DE methods which propose alternatives for the selection of parents during mutation and crossover. Section III-G shows recent improvement on population structure and operations.

A. Improving Mutation Strategy

Opposition-based DE (ODE) [26] and its improved algorithms (generalized opposition-based learning) [27], [28] utilize opposition number in population initialization, generation jumping, and local improvement [26]. Besides the ODE variants, there are diverse improvements on the mutation strategy. In differential covariance matrix adaptation (DCMA)-EA [29], the mutation operator of DE is combined with the mutation operator of CMA-ES [30], [31]. In [32], two-layer hierarchical DE uses modified DE/rand/1 and DE/current-to-best/1 strategies on the proposed layers. In Gaussian bare-bones differential evolution [33], the mutation strategy is improved by introducing a Gaussian random variable.

B. Improving Crossover Operator

Covariance matrix learning is presented to establish an appropriate coordinate system for the crossover operator [34]. In [35], DE is enhanced by mixing the binomial crossover operator and the eigenvector-based crossover operator. These methods have been shown effective especially on nonseparable functions with high conditioning. Besides providing an alternative coordinate system, the crossover operator can be improved by orthogonal crossover [36] which is based on orthogonal design.

C. Pooling Multiple Strategies

Some mutation strategies and crossover operators are suitable for rotated problems, and some others are useful for separable problems. To solve arbitrary problems, there are many works focusing on pooling multiple strategies, and simultaneously using them for each generation. Heterogeneous decentralized differential evolution [37] employs two mutation operators to generate new solutions. In composite differential evolution [38], the trial vectors are generated by three distinct mutation strategies. In self-adaptive multioperator differential evolution [39], the trial vectors are generated by four mutation strategies.

D. Strategy Adaptation

Based on pooling multiple strategies, the strategies can be adaptive by generation numbers or by selecting successful strategies. In jDElscop [40], two explorative strategies are used for the first half of the maximal number of generations. Then, an exploitative strategy is used with a small probability when the generation number exceeds the half of the maximal number of generations. On the other hand, there are many DE variants adapting strategies by selecting successful strategies, such as SaDE [16], EPSDE [18], self-adaptive strategy and parameter for differential evolution [41], strategy adaptation JADE [42], SaDE-modified multitrajectory search [43], and adaptive strategy selection-JADE [44]. In these methods, the successful strategies are used with high probability.

E. Adaptive Control Parameter

The performance of DE is sensitive to control parameters, such as scaling factor (F) and CR. In practical experience, there is no conclusive setting of control parameters that is the best for diverse problems [17]. To overcome this problem, adaptive or self-adaptive mechanisms are introduced to dynamically update the control parameters. For scaling factor adaptation, JADE [17] and other similar methods [19], [42], [45], [46] generate scaling factors according to a random variable distribution with a mean value that is learned by successful scaling factors in previous generations. For CR adaptation, SaDE [16] and other similar methods [19], [42]–[45] generate CRs at each generation according to a normal distribution with a mean value that is learned by successful CRs. In [15], [18], [33], [41], and [47]–[49], the control parameters are randomly generated in a pool. The successful control parameters have high probability to survive into the next generation, and

the unsuccessful control parameters are simply discarded or alter to other values. In FiADE [50], the control parameters are updated according to the dynamic of fitness values. Numerical comparisons of the adaptive DE variants are presented in [60] and [61].

F. Alternative for the Selection of Parents

DE can be enhanced by modifying the random selection of parents during mutation and crossover. In [19] and [51]–[55], the selection of parents is biased according to the fitness values of the solutions in the current population. The solutions with better fitness values gain more probabilities to be selected as parents. In [22], the selection of parents is based on the distance between the solutions in the current population. The solutions with small distance have higher probabilities to be selected as parents.

G. Population Structuring and Operations

The performance of DEs can be improved by specialized population structuring and operations. DEGL [15] uses a neighborhood-based mutation operator on a ring-topology of population. JADE [17] and SHADE [20] provide information about the progress direction by the use of archived inferior solutions. DDEM [58] migrates solutions in different subpopulations based on the affinity of the solutions. In [59], the population size is gradually reduced for improving the robustness of DE algorithms. In [62], an ageing strategy is employed to reinitialize a solution that stagnates in local minimum.

IV. Stagnation of DE

In this section, we investigate the stagnation of DE algorithms. The terminology, stagnation, expresses the following phenomena [56].

1) *The First Property of Stagnation:* The population has not converged to any fixed point.
2) *The Second Property of Stagnation:* The algorithm cannot find any better solutions.

When the stagnation happens to DE, the first property of stagnation can be shown by evaluating the average distance d_G between the target vectors and their centroid in the Gth generation

$$d_G = (1/\mathrm{NP}) \sum_{i=1}^{\mathrm{NP}} \left\| \vec{x}_{i,G} - \bar{x}_G \right\| \tag{14}$$

where $\bar{x}_G = (1/\mathrm{NP}) \sum_{i=1}^{\mathrm{NP}} \vec{x}_{i,G}$ is the centroid of the target vectors. If the sequence of the average distances does not decrease to a small value, it means that the population is not converged to a fixed point.

DE may converge the population to a fixed point when the optimization problem has only one local optimum, such as Bent Cigar function in CEC 2014 benchmarks. However, when the optimization problem has many local optima, such as the high-dimensional Rosenbrock's function in CEC 2014 benchmarks, DE may hardly converge the population to a fixed point even if the considered DE is very exploitative. Fig. 1 shows the average distances between target vectors and their centroid by running DE/best/1/bin on 30-D Bent Cigar function and Rosenbrock's function. After generation number 500, the distance begins to exponentially decrease to a small value on Bent Cigar function, but on Rosenbrock's function, the distance is not decreasing. Therefore, the population of DE/best/1/bin is not converged to a fixed point on the 30-D Rosenbrock's function.

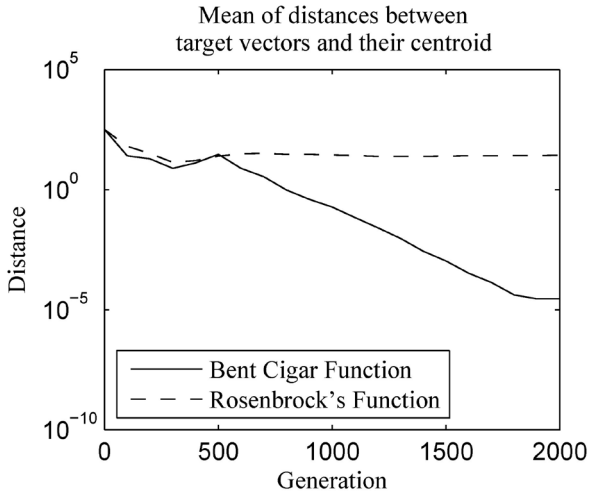


Fig. 1. Mean of distances between target vectors and their centroid by running DE/best/1/bin on 30-D Bent Cigar function and Rosenbrock's function.

The second property of stagnation states that DE cannot generate any successful solutions when the stagnation is happening. This phenomenon can be shown by evaluating the number of recently consecutive unsuccessful updates $q_{i,G}$ in the Gth generation

$$q_{i,G+1} = \begin{cases} 0, & \text{if } f\left(\vec{u}_{i,G}\right) \le f\left(\vec{x}_{i,G}\right) \\ q_{i,G} + 1, & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2, \ldots, \mathrm{NP} \tag{15}$$

with the initial values $q_{i,0} = 0$. If the sequence $\{q_{i,G}\}$ increases infinitely, it means that DE cannot generate any successful solutions for the ith target vector. Besides, it is also interesting to see the value of $q_{i,G}$ when a successful update is happening. In this moment, the value of $q_{i,G}$ is called a successful number of recently consecutive unsuccessful updates. To clearly illustrate the dynamics of this value, we define the mean value of the successful number of recently consecutive unsuccessful updates $\bar{q}_G^{(\mathrm{succ})}$ by

$$\bar{q}_G^{(\mathrm{succ})} = \left(\sum_{i=1}^{\mathrm{NP}} s_{i,G} q_{i,G} \right) \Big/ \left(\sum_{i=1}^{\mathrm{NP}} s_{i,G} \right) \tag{16}$$

where $s_{i,G}$ is an indicator of the successful update

$$s_{i,G} = \begin{cases} 1, & \text{if } f\left(\vec{u}_{i,G}\right) \le f\left(\vec{x}_{i,G}\right) \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

If the denominator of (16) is zero, we set $\bar{q}_G^{(\mathrm{succ})} = \bar{q}_{G-1}^{(\mathrm{succ})}$ to avoid undefined values. Fig. 2 shows the average numbers of recently consecutive unsuccessful updates $\bar{q}_G$ and their successful values $\bar{q}_G^{(\mathrm{succ})}$ by running DE/best/1/bin on 30-D Bent Cigar function and Rosenbrock's function.

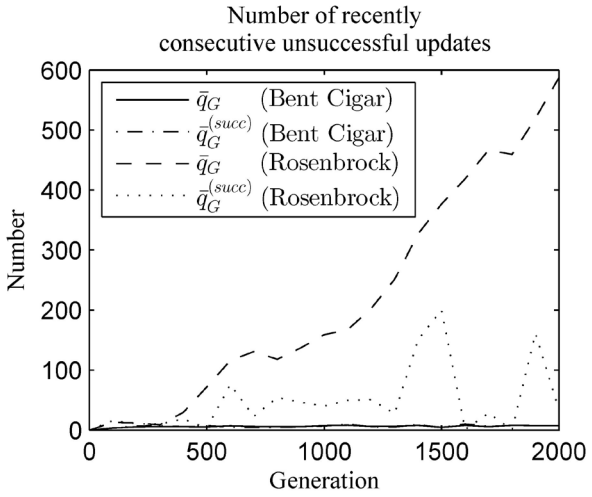


Fig. 2. Number of recently consecutive unsuccessful updates and successful number of recently consecutive unsuccessful updates by running DE/best/1/bin on 30-D Bent Cigar function and Rosenbrock's function.

For Bent Cigar function, the numbers of recently consecutive unsuccessful updates $\bar{q}_G$ are relatively small. It means that DE/best/1/bin has no problems on generating successful solutions, so that stagnation is not happening. For Rosenbrock's function, the numbers of recently consecutive unsuccessful updates $\bar{q}_G$ are generally increasing. It means that DE/best/1 has difficulties on generating successful solutions in the high-dimensional multimodal function. An interesting observation can be seen in the successful number of recently consecutive unsuccessful updates $\bar{q}_G^{(\text{succ})}$. For Rosenbrock's function, almost all successful values $\bar{q}_G^{(\text{succ})}$ are smaller than the average values $\bar{q}_G$. It means that DE easily generates successful solutions for those solutions that were recently updated.

This motivates our method to select parents from those solutions which are recently updated. In this way, DE is more likely to generate successful solutions, and thus accelerate convergence.

V. SPS Framework

As discussed in Section IV, it is possible to improve the performance of the parent-selecting method by utilizing the successful solutions. When stagnation is happening to DE, the parents should be selected from the recently updated solutions to increase the probabilities of generating successful solutions.

Assuming the population has not converged to a fixed point, we detect stagnation by examining which $q_{i,G}$ exceeds an unacceptable value, namely, Q. When stagnation is detected, the corresponding target vector $\vec{x}_{i,G}$ is called a stagnated solution, and the proposed SPS framework will select the parents from recently updated solutions. Specifically, the source of the parent $\vec{p}_{j,i,G}$ is given by

$$\vec{p}_{j,i,G} \in \begin{cases} \mathbf{P}_G, & \text{if } q_{i,G} \le Q \\ \mathbf{A}, & \text{otherwise.} \end{cases} \tag{18}$$

where $\mathbf{A}$ is the archive of recently updated solutions, and $Q \ge 0$ is a newly introduced stagnation tolerance on the number of consecutive unsuccessful updates to control the frequencies of the parents selected from the original population or the archive. This framework provides appropriate parents for objective functions that are best minimized by selecting parents from either the set of original population ($Q = \infty$) or the recently updated solutions ($Q = 0$), as well as generic functions that can be solved by interleaving both selection methods for parents.

The archive of recently updated solutions $\mathbf{A}$ is operated by a simple way to avoid considerable computation overhead. The archive is initiated to be a copy of the initial population. Then, after each generation, the successful solutions replace the oldest solutions to keep the archive size at NP.

Combining the proposed framework with DE, the SPS DE algorithm (SPS-DE for short) is presented. The pseudocode of DE/rand/1 algorithm with the proposed SPS framework is shown in Table II.

From Algorithm 2 in Table II, the selection of parents for mutation and crossover is modified by combining the original parent-selecting method with the proposed SPS method. Unlike the existed alternative for the selection of

TABLE II
Pseudocode of DE/rand/1 Algorithm With the Proposed SPS Framework

Algorithm 2 SPS-DE/rand/1/bin algorithm

Input: f: Objective function.
$MaxGen$: Maximal generation number.
$\vec{x}_{\min}$: Lower bounds.
$\vec{x}_{\max}$: Upper bounds.
NP: Population size.
F: Scaling factor.
CR: Crossover rate.
Q: Stagnation tolerance.

Initialize a population $\mathbf{P}_0 = \{\vec{x}_{1,0}, \ldots, \vec{x}_{NP,0}\}$ within the lower bounds $\vec{x}_{\min}$ and upper bounds $\vec{x}_{\max}$.
Set $q_{1,0} = \cdots = q_{NP,0} = 0$; $\mathbf{A} = \mathbf{P}_0$;
for $G = 0$ to $MaxGen$ **do**
  for $i = 1$ to NP **do**
    Randomly select $r_1 \ne r_2 \ne r_3$ from the set $I = \{1, 2, \ldots, NP\}$.
    if $q_{i,G} \le Q$
      Let $\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}$, and $\vec{p}_{4,i,G}$ be the r_1th, r_2th, r_3th, and ith vectors of $\mathbf{P}_G$, respectively.
    else
      Let $\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}$, and $\vec{p}_{4,i,G}$ be the r_1th, r_2th, r_3th, and ith vectors of $\mathbf{A}$, respectively.
    end if
    Generate a donor vector $\vec{v}_{i,G} = \text{mutate}_1(\vec{p}_{1,i,G}, \vec{p}_{2,i,G}, \vec{p}_{3,i,G}, F)$.
    Generate a trial vector $\vec{u}_{i,G} = \text{xover}(\vec{p}_{4,i,G}, \vec{v}_{i,G}, CR)$.
    if $f(\vec{u}_{i,G}) \le f(\vec{x}_{i,G})$
      $\vec{x}_{i,G+1} = \vec{u}_{i,G}$;
      $q_{i,G+1} = 0$;
      Replace the oldest solution in $\mathbf{A}$ with $\vec{u}_{i,G}$.
    else
      $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
      $q_{i,G+1} = q_{i,G} + 1$;
    end if
  end for
end for

Output: the candidate solution with the smallest objective function value.

parents [19], [22], [51]–[55], the proposed method is an adaptive approach that considers the feedback of the recently successful updates, so that the robustness of DE algorithm has potential to be enhanced by dynamically adapting promising parents for the situation of stagnation. In this way, DE should generate successful solutions more efficiently.

The proposed SPS framework can adequately cooperate with the archive of inferior solutions, which is used in JADE [17] and its variants [20], [42], [44], [45]. If a stagnated solution is detected, the proposed framework selects parents from the archive of recently updated solutions. Otherwise, DE generates solutions from the current population and the archive of inferior solutions. In this way, the information of recently updated solutions and inferior solutions provides DE complementary behavior, and thus has potential to improve the performance.

Moreover, the proposed framework is computationally efficient. The time complexities of the selection and replacement operations of the archive are $O(1)$ by implementing an array for the archive. Also, the counting of the consecutive unsuccessful updates and detection of stagnation do not increase the overall time complexity.

The source code of the proposed SPS-DE is available at https://github.com/SPS-DE/SPS-DE. In the source code, the implementation details are specifically written in MATLAB language. The simulation results in Section VI can be reproduced by running the codes. Furthermore, readers are able to integrate the proposed method into their DE variants by appropriately using the source code under the license.

VI. Simulation Results

In this section, we carry out a comprehensive evaluation and emphasize different aspects of the proposed SPS framework by dividing the presentation of simulation results into four subsections. Section VI-A incorporates the proposed SPS framework into two original DE algorithms and six state-of-the-art DE variants, and compares the performance of each DE algorithm with its "SPS" variant over four real-world optimization problems and 30 artificial benchmark functions. Section VI-B analyzes the selection of the stagnation tolerance (Q), and discusses its effect on the performance of DEs. In Section VI-C, we derive the relationship between the dimensions of the test functions and the performance of the incorporated DEs. Section VI-D provides an overall performance comparison to conclude the demonstration of all simulation results.

A. Improving the Performance of DE Algorithms

In this section, we incorporate the proposed SPS framework with two original DE algorithms and six state-of-the-art DE variants.

1) DE/rand/1/bin with $F = 0.7$, CR $= 0.5$.
2) DE/best/1/bin with $F = 0.7$, CR $= 0.5$.
3) DCMA-EA [29] with $Cr_m = 0.9$.
4) DEGL [15] with $\alpha = \beta = F = 0.7, Cr = 0.5$, and neighborhood size $= 0.1 \times$ NP.
5) DE with rank-based mutation (RBDE) [55] with $F = 0.7$, $Cr = 0.5$, and $\beta = 3.0$.
6) SaDE [16] with the learning period $LP = 50$.
7) JADE [17] with initial $\mu_F = 0.7$, $\mu_{\mathrm{CR}} = 0.5$, $c = 0.1$, $p = 0.05$, and the external archive.
8) SHADE [20], [21] with initial $M_F = \{0.7\}$, $M_{\mathrm{CR}} = \{0.5\}$, and $H =$ NP.

The common parameters of the incorporated algorithms are set as follows. The stagnation tolerance Q is set to 32 and the population size NP is $5 \times D$ in the problem dimension D. The initial population of all DE variants is uniformly distributed with the same random seeds over all the benchmark functions.

In the following experiments, numerical simulations are executed on four real-world optimization problems in CEC 2011 test suite [23] and 30 artificial benchmark functions in CEC 2014 competition [24]. For real-world optimization problems, we select four famous real-world problems from CEC 2011 test suite to judge the enhancement of the proposed SPS framework. The first real-world problem (rf_1) is an application to parameter estimation for frequency-modulated sound waves, which is the Problem no. 1 in CEC 2011. The second real-world problem (rf_2) is an application to spread spectrum radar poly-phase code design, which is the Problem no. 7 in CEC 2011. For the third real-world problem (rf_3), we select the 12th problem in CEC 2011, which is the so-called "Messenger" spacecraft trajectory optimization problem. Finally, the fourth real-world problem (rf_4) is selected from the 13th problem in CEC 2011, which defines the "Cassini 2" spacecraft trajectory optimization problem. For CEC 2014 benchmarks, we use all of the 30 artificial functions. Based on their characteristics, the functions of the CEC 2014 benchmark set can be divided into the following four classes. Functions $cf_1 - cf_3$ are unimodal, $cf_4 - cf_{16}$ are simple multimodal functions, $cf_{17} - cf_{22}$ are hybrid functions, and $cf_{23} - cf_{30}$ are composition functions.

The performance of DE is evaluated in terms of solution error measure and success rate (SR). The definition of solution error measure is given by $f(\vec{x}') - f(\vec{x}^*)$, where $\vec{x}^*$ is the global minimizer of the test function and $\vec{x}'$ is the best solution achieved after $D \times 10^4$ function evaluations. When the solution error measure is smaller than 10^{-8}, the evolution of this run is called a successful run, and the solution error measure will be reported as zero. On the other hand, the SR evaluates the probability of success of the DE algorithm. This measure computes the ratio between the number of successful runs and the total number of independent runs. Each algorithm is executed for 51 independent runs to obtain the mean and the standard deviation of the best-so-far errors, and evaluate the SR.

The paired Wilcoxon's rank-sum test is conducted at the 5% significance level to judge the significant difference of performance between the original DE algorithm and its SPS variant. The cases are marked with "+," "−," and "=" when the performance of the SPS variant is significantly better than, worse than, or similar to the DE algorithm without the proposed framework, respectively.

The smallest mean solution error is indicated with boldface font for each pair of the considered DE algorithms and their SPS variants. The best performance in terms of the mean solution error is underlined to highlight the best algorithm for each test function.

TABLE III
ERROR VALUES AND SRS OF THE ORIGINAL DE/RAND/1/BIN, DE/BEST/1/BIN, DCMA-EA, DEGL, AND THEIR CORRESPONDING SPS FRAMEWORK VARIANTS OVER FOUR REAL-WORLD OPTIMIZATION PROBLEMS AND THE 30-D CEC 2014 BENCHMARK FUNCTIONS

	DE/rand/1/bin			SPS-DE/rand/1/bin				DE/best/1/bin			SPS-DE/best/1/bin			
	Mean	St. D.	SR	Mean	St. D.	SR		Mean	St. D.	SR	Mean	St. D.	SR	
rf_1	**8.78E–01**	3.11E+00	86%	3.58E+00	5.79E+00	65%	–	**7.49E+00**	6.37E+00	39%	9.18E+00	7.94E+00	37%	=
rf_2	**1.70E+00**	1.02E–01	0%	1.71E+00	1.04E–01	0%	=	1.62E+00	1.15E–01	0%	**6.71E–01**	1.03E–01	0%	+
rf_3	2.47E+01	1.29E+00	0%	**1.58E+01**	7.53E–01	0%	+	1.52E+01	3.46E+00	0%	**1.44E+01**	2.53E+00	0%	=
rf_4	2.42E+01	2.04E+00	0%	**1.44E+01**	2.77E+00	0%	+	1.81E+01	3.41E+00	0%	**1.74E+01**	3.25E+00	0%	=
cf_1	1.00E+08	2.10E+07	0%	**4.31E+07**	1.12E+07	0%	+	2.46E+07	9.27E+06	0%	**1.02E+06**	5.35E+05	0%	+
cf_2	1.97E+05	3.73E+04	0%	**6.19E+04**	1.30E+04	0%	+	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=
cf_3	2.62E+02	3.76E+01	0%	**8.43E+01**	1.53E+01	0%	+	3.32E–05	2.00E–05	0%	**7.44E–06**	4.18E–06	0%	+
cf_4	1.39E+02	4.99E+00	0%	**8.12E+01**	1.54E+00	0%	+	7.68E+01	2.42E+01	0%	**7.13E+00**	1.88E+01	0%	+
cf_5	2.09E+01	4.66E–02	0%	2.09E+01	4.52E–02	0%	=	2.09E+01	4.56E–02	0%	2.09E+01	4.39E–02	0%	=
cf_6	3.09E+01	1.12E+00	0%	**2.72E+00**	7.05E–01	0%	+	**1.39E+00**	1.31E+00	0%	1.56E+00	1.20E+00	0%	=
cf_7	2.68E–01	1.42E–01	0%	**8.27E–03**	1.79E–03	0%	+	5.46E–03	6.61E–03	55%	**4.83E–03**	7.18E–03	59%	=
cf_8	1.27E+02	7.13E+00	0%	**9.94E+01**	8.47E+00	0%	+	8.90E+01	2.22E+01	0%	**2.06E+01**	5.41E+00	0%	+
cf_9	1.99E+02	1.17E+01	0%	**1.83E+02**	8.11E+00	0%	+	1.81E+02	1.12E+01	0%	**1.54E+02**	2.62E+01	0%	+
cf_{10}	4.10E+03	2.86E+02	0%	**2.86E+03**	2.77E+02	0%	+	1.71E+03	1.06E+03	0%	**2.74E+02**	1.68E+02	0%	+
cf_{11}	6.56E+03	3.22E+02	0%	**6.21E+03**	3.02E+02	0%	+	6.41E+03	2.93E+02	0%	**4.72E+03**	1.91E+03	0%	+
cf_{12}	2.12E+00	2.49E–01	0%	**8.57E–01**	3.55E–01	0%	+	2.09E+00	2.42E–01	0%	**7.18E–01**	3.58E–01	0%	+
cf_{13}	5.03E–01	4.69E–02	0%	**4.25E–01**	5.30E–02	0%	+	3.79E–01	4.26E–02	0%	**2.99E–01**	4.18E–02	0%	+
cf_{14}	2.85E–01	4.05E–02	0%	**2.74E–01**	2.91E–02	0%	=	3.90E–01	2.08E–01	0%	**2.94E–01**	1.37E–01	0%	+
cf_{15}	1.98E+01	1.26E+00	0%	**1.76E+01**	9.62E–01	0%	+	1.66E+01	1.26E+00	0%	**1.49E+01**	9.23E–01	0%	+
cf_{16}	1.25E+01	2.25E–01	0%	**1.20E+01**	2.50E–01	0%	+	1.22E+01	2.73E–01	0%	**1.14E+01**	3.08E–01	0%	+
cf_{17}	3.13E+06	6.82E+05	0%	**9.18E+05**	3.74E+05	0%	+	7.64E+05	2.61E+05	0%	**3.78E+04**	2.02E+04	0%	+
cf_{18}	7.22E+04	2.82E+04	0%	**7.89E+02**	9.54E+02	0%	+	**1.68E+03**	1.28E+03	0%	5.19E+03	5.82E+03	0%	–
cf_{19}	1.13E+01	4.66E–01	0%	**5.70E+00**	2.88E–01	0%	+	**6.29E+00**	1.26E+00	0%	6.70E+00	1.01E+01	0%	+
cf_{20}	1.05E+03	2.92E+02	0%	**3.02E+02**	7.58E+01	0%	+	1.14E+02	1.65E+01	0%	**8.05E+01**	1.76E+01	0%	+
cf_{21}	3.00E+05	1.07E+05	0%	**6.87E+04**	3.20E+04	0%	+	3.45E+04	1.49E+04	0%	**8.75E+03**	7.85E+03	0%	+
cf_{22}	2.64E+02	8.14E+01	0%	**6.70E+01**	6.72E+01	0%	+	**1.55E+02**	1.10E+02	0%	2.06E+02	1.37E+02	0%	=
cf_{23}	3.15E+02	3.25E–03	0%	3.15E+02	4.15E–04	0%	+	3.15E+02	4.02E–13	0%	3.15E+02	4.02E–13	0%	=
cf_{24}	2.21E+02	2.38E+00	0%	**2.04E+02**	3.32E–01	0%	+	**2.23E+02**	7.13E+00	0%	2.25E+02	5.34E+00	0%	=
cf_{25}	2.25E+02	2.66E+00	0%	**2.11E+02**	2.04E+00	0%	+	2.09E+02	1.77E+00	0%	**2.03E+02**	8.71E–01	0%	+
cf_{26}	1.00E+02	5.18E–02	0%	1.00E+02	4.58E–02	0%	+	1.00E+02	6.13E–02	0%	1.00E+02	4.26E–02	0%	+
cf_{27}	5.43E+02	5.74E+01	0%	**3.27E+02**	6.30E+00	0%	+	**3.45E+02**	4.44E+01	0%	3.49E+02	4.05E+01	0%	=
cf_{28}	1.00E+03	2.20E+01	0%	**8.03E+02**	1.74E+01	0%	+	**7.74E+02**	1.03E+02	0%	7.90E+02	5.58E+01	0%	–
cf_{29}	1.53E+04	3.58E+03	0%	**2.20E+03**	2.98E+02	0%	+	3.57E+03	1.97E+03	0%	**1.22E+03**	2.76E+02	0%	+
cf_{30}	7.70E+03	1.13E+03	0%	**2.22E+03**	5.82E+02	0%	+	2.24E+03	6.54E+02	0%	**1.62E+03**	7.68E+02	0%	+
Total number of (+/=/–): 30/3/1								Total number of (+/=/–): 21/11/2						

	DCMA-EA			SPS-DCMA-EA				DEGL			SPS-DEGL			
	Mean	St. D.	SR	Mean	St. D.	SR		Mean	St. D.	SR	Mean	St. D.	SR	
rf_1	**9.08E+00**	5.16E+00	12%	1.34E+01	7.34E+00	18%	–	**7.67E+00**	5.86E+00	6%	9.10E+00	6.74E+00	31%	=
rf_2	1.63E+00	9.68E–02	0%	**6.26E–01**	1.11E–01	0%	+	1.27E+00	1.16E–01	0%	**9.84E–01**	1.53E–01	0%	+
rf_3	2.11E+01	1.27E+00	0%	**1.48E+01**	1.16E+00	0%	+	1.75E+01	2.14E+00	0%	**1.48E+01**	2.12E+00	0%	+
rf_4	2.30E+01	1.82E+00	0%	**1.61E+01**	1.91E+00	0%	+	2.30E+01	1.41E+00	0%	**2.14E+01**	1.48E+00	0%	+
cf_1	1.12E+06	4.69E+05	0%	**4.70E+05**	2.07E+05	0%	+	4.96E+05	4.24E+05	0%	**4.61E+05**	6.03E+05	0%	=
cf_2	**1.18E–09**	5.11E–09	94%	2.80E–09	6.58E–09	82%	=	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=
cf_3	4.92E–06	9.56E–06	0%	**9.04E–07**	1.72E–06	0%	+	4.51E–07	1.71E–06	88%	**1.46E–07**	6.36E–07	88%	=
cf_4	5.47E+01	2.33E+01	0%	**2.11E+01**	1.85E+01	0%	+	**4.86E+01**	3.37E+01	0%	5.53E+01	3.30E+01	0%	=
cf_5	2.09E+01	4.72E–02	0%	2.09E+01	5.49E–02	0%	=	2.08E+01	5.26E–02	0%	**2.00E+01**	7.52E–05	0%	+
cf_6	3.52E–01	7.15E–01	2%	**2.28E–01**	6.90E–01	0%	+	**5.21E+00**	1.94E+00	0%	5.70E+00	1.83E+00	0%	=
cf_7	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=	1.12E–02	1.30E–02	33%	**9.21E–03**	1.47E–02	57%	=
cf_8	7.93E+01	5.60E+00	0%	**8.10E+00**	2.65E+00	0%	+	4.47E+01	1.02E+01	0%	**2.76E+01**	7.24E+00	0%	+
cf_9	1.31E+02	9.54E+00	0%	**1.21E+01**	3.25E+00	0%	+	6.65E+01	1.35E+01	0%	**5.24E+01**	1.41E+01	0%	+
cf_{10}	3.19E+03	3.37E+02	0%	**8.05E+01**	8.56E+01	0%	+	2.56E+03	3.93E+02	0%	**7.52E+02**	4.21E+02	0%	+
cf_{11}	6.00E+03	2.91E+02	0%	**6.06E+02**	3.97E+02	0%	+	4.24E+03	4.36E+02	0%	**2.79E+03**	5.15E+02	0%	+
cf_{12}	1.89E+00	2.56E–01	0%	**1.32E+00**	3.18E–01	0%	+	1.37E+00	1.94E–01	0%	**1.23E–01**	5.71E–02	0%	+
cf_{13}	2.45E–01	2.94E–02	0%	**1.29E–01**	2.15E–02	0%	+	2.62E–01	5.30E–02	0%	**1.98E–01**	4.35E–02	0%	+
cf_{14}	**2.99E–01**	3.14E–02	0%	3.19E–01	3.69E–02	0%	–	**2.26E–01**	4.24E–02	0%	2.35E–01	3.96E–02	0%	=
cf_{15}	1.24E+01	9.27E–01	0%	**3.24E+00**	6.23E–01	0%	+	9.09E+00	2.08E+00	0%	**5.14E+00**	1.70E+00	0%	+
cf_{16}	1.19E+01	2.43E–01	0%	**1.01E+01**	4.11E–01	0%	+	1.15E+01	3.78E–01	0%	**1.04E+01**	6.90E–01	0%	+
cf_{17}	9.16E+04	3.71E+04	0%	**2.45E+04**	1.32E+04	0%	+	1.07E+05	6.52E+04	0%	**9.69E+04**	7.31E+04	0%	=
cf_{18}	3.27E+02	6.13E+01	0%	**1.77E+02**	1.54E+02	0%	+	**3.70E+02**	2.42E+02	0%	6.28E+02	5.08E+02	0%	=
cf_{19}	6.25E+00	7.74E–01	0%	**4.81E+00**	6.71E–01	0%	+	1.44E+01	1.62E+01	0%	**1.04E+01**	1.13E+01	0%	+
cf_{20}	9.07E+01	1.17E+01	0%	**6.00E+01**	1.54E+01	0%	+	9.92E+01	2.18E+01	0%	**9.78E+01**	3.54E+01	0%	=
cf_{21}	5.51E+03	1.36E+03	0%	**3.77E+03**	1.53E+03	0%	+	7.30E+03	3.98E+03	0%	**6.98E+03**	6.79E+03	0%	=
cf_{22}	2.24E+02	4.59E+01	0%	**1.53E+02**	4.38E+01	0%	+	**1.83E+02**	5.56E+01	0%	2.18E+02	7.92E+01	0%	=
cf_{23}	3.15E+02	4.02E–13	0%	3.15E+02	4.02E–13	0%	=	3.15E+02	3.98E–13	0%	3.15E+02	3.94E–13	0%	=
cf_{24}	2.24E+02	7.10E–01	0%	2.24E+02	5.89E–01	0%	+	**2.25E+02**	1.65E+00	0%	2.27E+02	4.15E+00	0%	–
cf_{25}	2.06E+02	3.49E+00	0%	2.06E+02	3.53E+00	0%	=	2.10E+02	2.01E+00	0%	**2.08E+02**	3.83E+00	0%	=
cf_{26}	1.00E+02	2.11E–01	0%	1.00E+02	3.58E–02	0%	+	1.00E+02	3.63E–02	0%	1.00E+02	4.34E–02	0%	+
cf_{27}	**3.00E+02**	4.74E–04	0%	3.01E+02	3.79E+00	0%	=	**4.38E+02**	5.64E+01	0%	4.40E+02	4.36E+01	0%	=
cf_{28}	**8.38E+02**	4.02E+01	0%	8.40E+02	2.59E+01	0%	=	**8.78E+02**	8.16E+01	0%	9.22E+02	1.05E+02	0%	–
cf_{29}	3.87E+03	9.54E+02	0%	**1.48E+03**	1.74E+02	0%	+	1.41E+03	4.78E+02	0%	**1.12E+03**	2.65E+02	0%	+
cf_{30}	**1.80E+03**	2.56E+02	0%	1.81E+03	3.22E+02	0%	=	2.06E+03	5.27E+02	0%	**2.02E+03**	4.47E+02	0%	=
Total number of (+/=/–): 24/8/2								Total number of (+/=/–): 15/17/2						

Tables III and IV show the results on four real-world optimization problems in CEC 2011 test suite and 30-D test functions in CEC 2014 benchmarks. The results reveal that the SPS framework significantly improves the performance of DE algorithms for both explorative and exploitative strategies, DE/rand/1/bin and DE/best/1/bin, respectively. For DE/rand/1/bin, the proposed framework advances significant improvement or similar performance

TABLE IV
ERROR VALUES AND SRS OF THE ORIGINAL JADE, RBDE, SADE, SHADE, AND THEIR CORRESPONDING SPS FRAMEWORK VARIANTS OVER FOUR REAL-WORLD OPTIMIZATION PROBLEMS AND THE 30-D CEC 2014 BENCHMARK FUNCTIONS

	JADE			SPS-JADE				RBDE			SPS-RBDE			
	Mean	St. D.	SR	Mean	St. D.	SR		Mean	St. D.	SR	Mean	St. D.	SR	
rf_1	**3.54E+00**	5.43E+00	41%	5.48E+00	6.35E+00	55%	=	**1.95E+00**	4.48E+00	76%	5.22E+00	6.30E+00	55%	–
rf_2	1.15E+00	8.03E–02	0%	**7.73E–01**	1.65E–01	0%	+	**1.70E+00**	9.61E–02	0%	1.73E+00	9.25E–02	0%	=
rf_3	1.72E+01	1.11E+00	0%	**1.48E+01**	1.42E+00	0%	+	2.07E+01	1.93E+00	0%	**1.53E+01**	8.95E–01	0%	+
rf_4	1.61E+01	2.13E+00	0%	**1.37E+01**	2.86E+00	0%	+	1.46E+01	4.86E+00	0%	**1.38E+01**	3.00E+00	0%	=
cf_1	**1.73E+02**	6.49E+02	0%	3.10E+02	1.24E+03	0%	=	7.22E+07	1.52E+07	0%	**2.51E+07**	7.54E+06	0%	+
cf_2	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=	2.45E+00	9.03E–01	0%	**1.27E+00**	4.54E–01	0%	+
cf_3	**1.28E+01**	1.08E+01	4%	1.45E+01	1.64E+01	2%	=	9.49E–01	2.55E–01	0%	**2.78E–01**	7.97E–02	0%	+
cf_4	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=	8.08E+01	5.12E+00	0%	**7.13E+01**	5.16E–01	0%	+
cf_5	2.03E+01	4.02E–02	0%	**2.02E+01**	7.81E–02	0%	+	2.09E+01	5.43E–02	0%	2.09E+01	5.18E–02	0%	=
cf_6	1.25E+01	9.96E–01	0%	**2.23E+00**	1.38E+00	0%	+	2.57E+01	3.13E+00	0%	**1.72E–01**	1.56E–01	0%	+
cf_7	7.00E–10	5.00E–09	98%	**0.00E+00**	0.00E+00	100%	=	5.99E–07	8.55E–07	0%	**6.08E–08**	2.61E–08	0%	+
cf_8	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=	1.20E+02	7.42E+00	0%	**7.58E+01**	7.76E+00	0%	+
cf_9	3.67E+01	5.14E+00	0%	**2.24E+01**	6.32E+00	0%	+	1.90E+02	1.00E+01	0%	**1.73E+02**	9.03E+00	0%	+
cf_{10}	**4.61E–03**	4.35E–03	0%	1.24E–02	1.73E–02	0%	+	3.82E+03	3.13E+02	0%	**1.87E+03**	2.87E+02	0%	+
cf_{11}	2.15E+03	2.57E+02	0%	**1.41E+03**	3.22E+02	0%	+	6.50E+03	2.78E+02	0%	**6.21E+03**	3.19E+02	0%	+
cf_{12}	4.15E–01	5.26E–02	0%	**1.01E–01**	4.54E–02	0%	+	2.08E+00	2.31E–01	0%	**7.63E–01**	3.96E–01	0%	+
cf_{13}	2.15E–01	3.21E–02	0%	**1.16E–01**	3.22E–02	0%	+	4.57E–01	3.96E–02	0%	**3.92E–01**	5.66E–02	0%	+
cf_{14}	**2.20E–01**	2.76E–02	0%	2.85E–01	4.18E–02	0%	–	2.77E–01	3.05E–02	0%	**2.60E–01**	3.05E–02	0%	+
cf_{15}	4.08E+00	4.06E–01	0%	**2.63E+00**	5.68E–01	0%	+	1.80E+01	1.08E+00	0%	**1.66E+01**	1.06E+00	0%	+
cf_{16}	9.71E+00	3.34E–01	0%	**8.37E+00**	7.73E–01	0%	+	1.24E+01	1.90E–01	0%	**1.18E+01**	2.70E–01	0%	+
cf_{17}	**1.77E+04**	1.20E+05	0%	3.78E+04	9.62E+04	0%	=	1.71E+06	4.92E+05	0%	**4.62E+05**	2.24E+05	0%	+
cf_{18}	4.74E+02	2.23E+03	0%	**4.11E+01**	5.14E+01	0%	=	4.20E+03	3.37E+03	0%	**1.08E+03**	1.91E+03	0%	+
cf_{19}	4.53E+00	1.05E+00	0%	**3.69E+00**	5.05E–01	0%	+	6.78E+00	1.29E+00	0%	**5.09E+00**	4.86E–01	0%	+
cf_{20}	3.38E+03	1.73E+03	0%	**3.08E+03**	2.07E+03	0%	=	1.89E+02	2.33E+01	0%	**1.10E+02**	1.50E+01	0%	+
cf_{21}	5.44E+04	8.28E+04	0%	**2.13E+04**	4.31E+04	0%	=	8.78E+04	2.45E+04	0%	**2.00E+04**	1.22E+04	0%	+
cf_{22}	1.59E+02	6.95E+01	0%	**1.34E+02**	1.04E+02	0%	=	1.80E+02	6.09E+01	0%	**1.13E+02**	1.27E+02	0%	+
cf_{23}	3.15E+02	4.02E–13	0%	3.15E+02	4.02E–13	0%	=	3.15E+02	1.66E–07	0%	3.15E+02	4.58E–08	0%	+
cf_{24}	2.25E+02	2.14E+00	0%	2.25E+02	1.57E+00	0%	=	2.03E+02	3.78E+00	0%	**2.01E+02**	1.27E–01	0%	+
cf_{25}	2.04E+02	1.07E+00	0%	2.04E+02	8.31E–01	0%	=	2.20E+02	2.61E+00	0%	**2.07E+02**	1.50E+00	0%	+
cf_{26}	1.00E+02	3.32E–02	0%	1.00E+02	3.14E–02	0%	+	1.00E+02	4.46E–02	0%	1.00E+02	4.64E–02	0%	+
cf_{27}	3.15E+02	3.54E+01	0%	**3.07E+02**	2.49E+01	0%	+	3.03E+02	8.49E–01	0%	3.03E+02	7.39E+00	0%	+
cf_{28}	8.07E+02	2.91E+01	0%	**7.75E+02**	3.55E+01	0%	+	8.43E+02	6.94E+01	0%	**7.98E+02**	2.20E+01	0%	+
cf_{29}	**7.36E+02**	1.20E+02	0%	7.61E+02	2.09E+02	0%	=	7.00E+03	1.96E+03	0%	**1.63E+03**	2.57E+02	0%	+
cf_{30}	1.39E+03	6.23E+02	0%	**1.33E+03**	4.98E+02	0%	=	3.65E+03	5.82E+02	0%	**1.33E+03**	4.98E+02	0%	+
				Total number of (+/=/–): 16/17/1							Total number of (+/=/–): 30/3/1			

	SaDE			SPS-SaDE				SHADE			SPS-SHADE			
	Mean	St. D.	SR	Mean	St. D.	SR		Mean	St. D.	SR	Mean	St. D.	SR	
rf_1	**7.29E–01**	2.54E+00	92%	1.68E+00	4.30E+00	86%	=	**3.15E+00**	5.03E+00	31%	4.37E+00	5.82E+00	63%	=
rf_2	1.44E+00	1.04E–01	0%	**6.93E–01**	1.10E–01	0%	+	1.17E+00	7.70E–02	0%	**6.00E–01**	8.92E–02	0%	+
rf_3	1.87E+01	2.07E+00	0%	**1.52E+01**	8.50E–01	0%	+	1.64E+01	9.76E–01	0%	**1.54E+01**	1.04E+00	0%	+
rf_4	**1.66E+01**	3.49E+00	0%	1.93E+01	2.37E+00	0%	–	1.61E+01	2.63E+00	0%	**1.54E+01**	3.72E+00	0%	=
cf_1	**1.43E+05**	9.27E+04	0%	1.48E+05	1.12E+05	0%	=	**9.95E+02**	2.10E+03	0%	1.04E+03	2.45E+03	0%	=
cf_2	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=
cf_3	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=
cf_4	5.94E+00	1.84E+01	0%	**4.66E+00**	1.56E+01	0%	=	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=
cf_5	2.09E+01	4.50E–02	0%	2.09E+01	5.60E–02	0%	=	2.03E+01	3.33E–02	0%	**2.02E+01**	8.85E–02	0%	+
cf_6	**2.02E–01**	4.95E–01	10%	2.81E–01	5.10E–01	18%	=	6.15E+00	4.31E+00	25%	**3.68E–02**	1.56E–01	94%	+
cf_7	2.90E–04	2.07E–03	98%	**0.00E+00**	0.00E+00	100%	=	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=
cf_8	1.27E+01	2.64E+00	0%	**7.80E–01**	7.79E–01	31%	+	0.00E+00	0.00E+00	100%	0.00E+00	0.00E+00	100%	=
cf_9	1.03E+02	1.75E+01	0%	**2.00E+01**	5.85E+00	0%	+	2.77E+01	4.04E+00	0%	**1.70E+01**	5.82E+00	0%	+
cf_{10}	7.20E+02	1.28E+02	0%	**2.38E+02**	6.93E+01	0%	+	1.57E–01	3.90E–02	0%	**3.45E–02**	2.42E–02	0%	+
cf_{11}	5.84E+03	3.18E+02	0%	**1.61E+03**	6.12E+02	0%	+	1.98E+03	1.96E+02	0%	**1.18E+03**	3.79E+02	0%	+
cf_{12}	1.89E+00	2.18E–01	0%	**1.49E+00**	3.88E–01	0%	+	3.17E–01	3.70E–02	0%	**8.67E–02**	2.79E–02	0%	+
cf_{13}	2.76E–01	3.21E–02	0%	**1.85E–01**	2.51E–02	0%	+	2.17E–01	2.85E–02	0%	**8.15E–02**	2.23E–02	0%	+
cf_{14}	**2.52E–01**	2.64E–02	0%	2.60E–01	3.22E–02	0%	=	**2.12E–01**	2.38E–02	0%	2.60E–01	3.50E–02	0%	–
cf_{15}	1.09E+01	9.64E–01	0%	**2.93E+00**	7.30E–01	0%	+	3.85E+00	4.05E–01	0%	**2.91E+00**	8.13E–01	0%	+
cf_{16}	1.22E+01	2.10E–01	0%	**1.09E+01**	3.62E–01	0%	+	9.59E+00	3.13E–01	0%	**8.14E+00**	7.78E–01	0%	+
cf_{17}	1.19E+03	4.06E+02	0%	**1.17E+03**	3.19E+02	0%	=	**6.75E+02**	2.73E+02	0%	7.39E+02	3.22E+02	0%	=
cf_{18}	6.86E+01	1.71E+01	0%	**6.72E+01**	1.41E+01	0%	=	**1.37E+01**	7.55E+00	0%	1.55E+01	1.03E+01	0%	=
cf_{19}	4.89E+00	5.32E–01	0%	**3.41E+00**	7.68E–01	0%	+	3.99E+00	6.06E–01	0%	**2.77E+00**	7.68E–01	0%	+
cf_{20}	**2.26E+01**	9.61E+00	0%	2.59E+01	1.33E+01	0%	=	**5.42E+00**	2.16E+00	0%	5.54E+00	2.32E+00	0%	=
cf_{21}	3.41E+02	1.19E+02	0%	**3.36E+02**	1.22E+02	0%	=	1.76E+02	8.97E+01	0%	**1.64E+02**	9.40E+01	0%	=
cf_{22}	1.35E+02	5.29E+01	0%	**8.33E+01**	6.77E+01	0%	+	**9.53E+01**	4.85E+01	0%	1.01E+02	8.74E+01	0%	=
cf_{23}	3.15E+02	4.02E–13	0%	3.15E+02	4.02E–13	0%	=	3.15E+02	4.02E–13	0%	3.15E+02	4.02E–13	0%	=
cf_{24}	2.24E+02	8.19E–01	0%	2.24E+02	1.83E+00	0%	=	2.23E+02	8.11E–01	0%	2.23E+02	9.71E–01	0%	=
cf_{25}	**2.03E+02**	2.93E+00	0%	2.04E+02	2.90E+00	0%	=	2.04E+02	7.38E–01	0%	2.04E+02	8.37E–01	0%	=
cf_{26}	1.00E+02	3.07E–02	0%	1.00E+02	3.02E–02	0%	+	1.00E+02	3.20E–02	0%	1.00E+02	1.72E–02	0%	+
cf_{27}	3.44E+02	4.99E+01	0%	**3.26E+02**	4.25E+01	0%	=	3.08E+02	2.73E+01	0%	**3.01E+02**	5.21E+00	0%	=
cf_{28}	8.42E+02	3.44E+01	0%	**8.33E+02**	2.87E+01	0%	=	**7.90E+02**	2.76E+01	0%	8.00E+02	3.92E+01	0%	–
cf_{29}	**7.76E+02**	6.72E+01	0%	7.93E+02	8.76E+01	0%	=	7.20E+02	8.59E+00	0%	**7.18E+02**	3.79E+00	0%	=
cf_{30}	**8.27E+02**	2.79E+02	0%	8.42E+02	2.63E+02	0%	=	**1.15E+03**	3.51E+02	0%	1.22E+03	3.27E+02	0%	=
				Total number of (+/=/–): 13/20/1							Total number of (+/=/–): 13/19/2			

except for rf_1 function. For DE/best/1/bin, the proposed SPS framework reveals either similar or significant performance improvement in 32 of 34 functions with the exception of cf_{18} and cf_{28} functions. The considered state-of-the-art DEs are also enhanced by the proposed framework. The incorporation of the proposed framework exhibits either similar or significantly superior performance in 32 of 34 functions for SPS-DCMA-EA, SPS-DEGL, and SPS-SHADE. The other

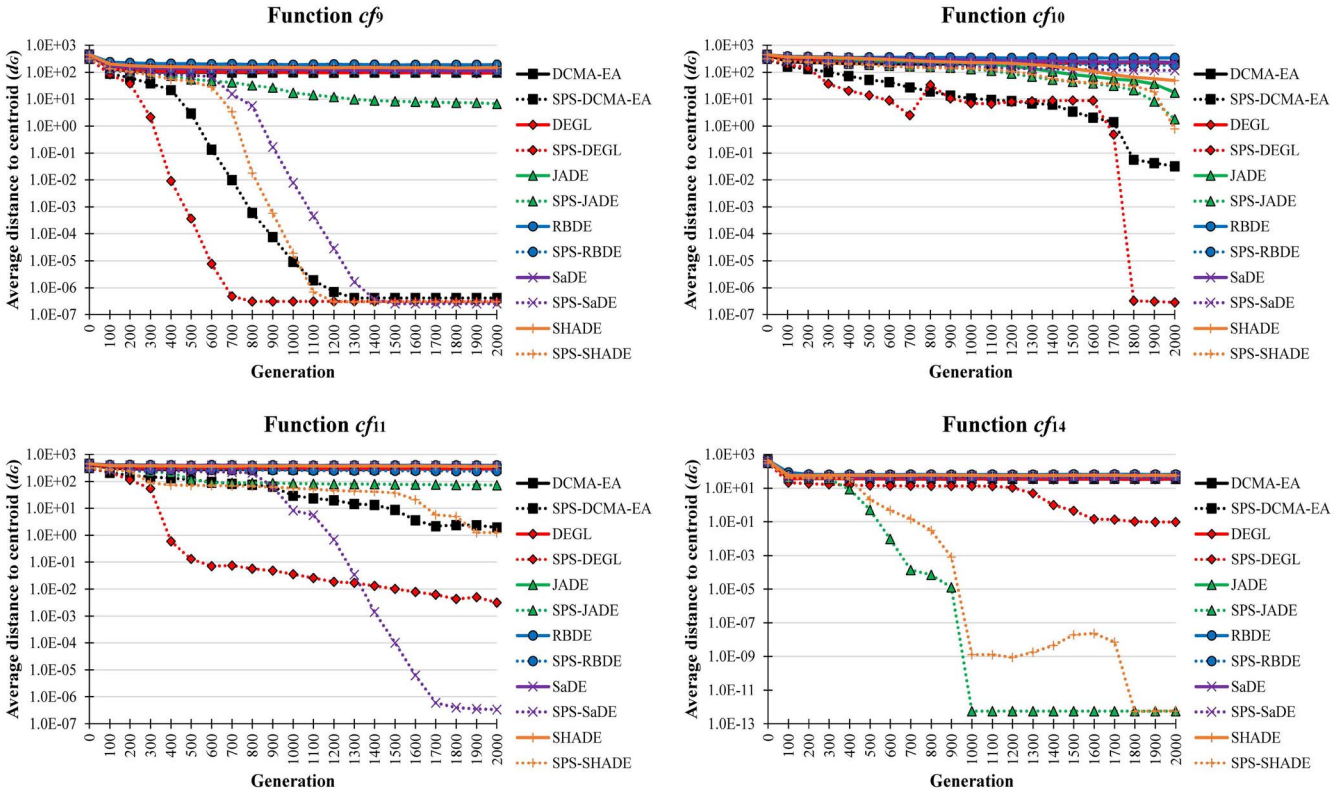


Fig. 3. Contraction graph (median curves) for the DE variants on the 30-D cf_9, cf_{10}, cf_{11}, and cf_{14} in CEC 2014 benchmark functions over 51 independent runs.

incorporated DEs (SPS-JADE, SPS-RBDE, and SPS-SaDE) reveal either significantly better or equal performance in 33 of 34 functions.

For the multimodal functions cf_9, cf_{10}, cf_{11}, cf_{12}, cf_{13}, cf_{15}, cf_{16}, cf_{19}, and cf_{26}, the proposed framework significantly enhances the performance for all the considered DE algorithms. In other test functions, the majority of the SPS-DEs exhibit similar or superior performance with the exception of the six-dimensional rf_1 function. For such low-dimensional function, increasing the value of the stagnation tolerance (Q) may significantly improve the performance of SPS-DEs as shown in Section VI-C.

Fig. 3 shows contraction graphs for the test functions cf_9, cf_{10}, cf_{11}, and cf_{14}. Generally, the proposed SPS framework provides fast convergence on the distance between the target vectors and their centroid. For the functions cf_9 and cf_{10}, SPS-DEGL decreases the solution distance faster than the other DE algorithms. For the function cf_{11}, SPS-DEGL exhibits the fastest contraction speed at generation number 300. Then, SPS-SaDE overtakes SPS-DEGL after generation number 1400, and SPS-SaDE exponentially decreases the distance to a relatively small value. For the function cf_{14}, SPS-JADE and SPS-SHADE decrease the distance fastest. Overall, in these functions, the incorporated DE algorithms converge the solutions to a fixed point more efficiently, and thus prevent the first property of stagnation described in Section IV.

Fig. 4 shows four graphs for the average numbers of the consecutive unsuccessful updates ($\bar{q}_G$) over the 30-D cf_9, cf_{10}, cf_{11}, and cf_{14}. The average numbers of the consecutive unsuccessful updates have a high correlation with the distance between the target vectors and their centroid. Fig. 3 shows that the solution distance may occasionally not decrease anymore when the solution distance has been decreased to an extremely small value. Then, Fig. 4 shows that, in this moment, the number of consecutive unsuccessful updates may dramatically increase. For instance, SPS-DEGL, SPS-DCMA-EA, SPS-SaDE, and SPS-SHADE converge the solution distances to the values between 10^{-6} and 10^{-7} in the graph of the function cf_9. Meanwhile, their numbers of consecutive unsuccessful updates are dramatically increasing as shown in Fig. 4. In this situation, the solutions are converged to a fixed point, and DE algorithms cannot generate any better solutions. Similar phenomena can be observed in the other functions, for example, SPS-DEGL in cf_{10}, SPS-SaDE in cf_{11}, and SPS-JADE in cf_{14}. Without considering the above special cases, Fig. 4 shows that SPS-DEs exhibit smaller numbers of consecutive unsuccessful updates than those of the original DE algorithms. Therefore, the proposed framework provides more successful updates and thus prevents the second property of stagnation described in Section IV.

To quantify the overhead of the proposed SPS framework in real computation time, we measure the running time of all the considered DE algorithms T_1 and those of their SPS variants T_2 on all CEC 2014 benchmark functions at $D = 30$. The computational overhead of the proposed framework is then estimated by $(T_2 - T_1)/T_1 = 6.08\%$, which shows that the implementation of the proposed framework is time-efficient. Therefore, the proposed method provides significant improvement in solution error measure and with a relatively small overhead.

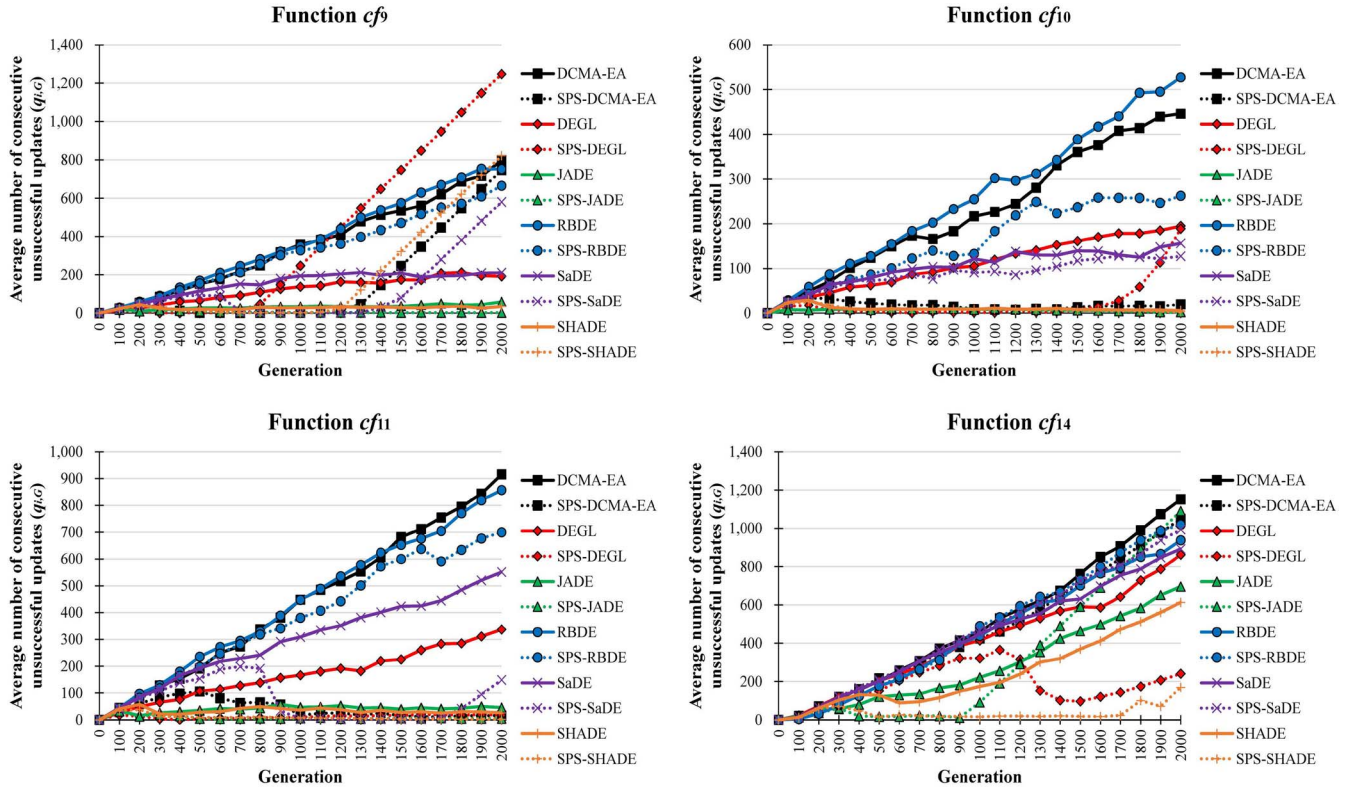


Fig. 4. Graph of the average numbers of consecutive unsuccessful updates (median curves of $\bar{q}_G$) for the DE variants on the 30-D cf_9, cf_{10}, cf_{11}, and cf_{14} in CEC 2014 benchmark functions over 51 independent runs.

B. Effect of the Stagnation Tolerance

The proposed SPS framework introduces a new parameter, namely, stagnation tolerance (Q). The varied selection of the stagnation tolerance (Q) greatly affects the population diversity, which is an unavoidable problem to balance between exploration and exploitation for DE algorithms. Nevertheless, some empirical guidelines can be provided since the framework becomes no operations if the value of the stagnation tolerance (Q) approaches infinity. Large stagnation tolerance (Q) restricts the exploitation power, and the incorporated DE algorithm may converge slowly when locating the global optimum for a difficult optimization problem. On the other hand, if the value of Q is close to zero, the framework always selects parents from the recently updated solutions. The incorporated algorithm may lose the population diversity, and suffer from the problem of convergence premature.

To reveal the overall significant difference of the DE algorithms for a stagnation tolerance (Q) varying from 1 to 512, we use the "positive subtracts negative" value (P–N value) [35] to evaluate the performance gain of the incorporated DE algorithm for a certain stagnation tolerance. Specifically, the P–N value is given by

$$\text{POSITIVE}–\text{NEGATIVE} \tag{19}$$

where POSITIVE and NEGATIVE are the total numbers of the cases that the performance of the SPS variant is significantly better than or worse than the original DE algorithm in CEC 2014 functions at $D = 30$, respectively. The P–N value reveals how many the significantly improved cases exceed the significantly weakened cases. For example, Table III shows that the total numbers of the positive markers (+) and the negative markers (−) are respectively, 20 and 2, for DE/best/1/bin without considering the real-world optimization problems (rf_1, rf_2, rf_3, and rf_4). In this case, the P–N value is therefore $20 - 2 = 18$.

TABLE V
P–N VALUES OF DE/RAND/1/BIN, DE/BEST/1/BIN, DCMA-EA, DEGL, JADE, RBDE, SADE, AND SHADE INCORPORATED WITH THE PROPOSED SPS FRAMEWORK WITH INCREASING VALUES OF THE STAGNATION TOLERANCE (Q) OVER CEC 2014 BENCHMARK FUNCTIONS AT $D = 30$

Algorithm \ Q	1	2	4	8	16	32	64	128	256	512
DE/rand/1/bin	28	28	28	28	**29**	28	**29**	27	24	15
DE/best/1/bin	15	15	17	16	17	**18**	17	**18**	17	13
DCMA-EA	11	14	15	15	**20**	**20**	16	15	11	2
DEGL	7	5	8	8	10	10	12	**13**	10	8
JADE	–8	–5	–1	6	10	**12**	10	10	5	1
RBDE	27	28	**29**	28	28	**29**	27	23	21	15
SaDE	2	1	3	6	**12**	11	11	11	10	3
SHADE	–4	–2	–3	–1	5	9	**12**	10	7	4
Average	9.75	10.5	12	13.25	16.375	**17.125**	16.75	15.875	13.125	7.625

Table V shows P–N values of the considered DE algorithms for a stagnation tolerance (Q) exponentially varying from 1 to 512. The best choice of the stagnation tolerance (Q) is marked by bold font for each DE algorithm. Generally, the illustrated P–N values demonstrate the fact that the most significant improvement appears in the intermediate values of the stagnation tolerance. More specifically, the highest P–N values can be obtained with the stagnation tolerance

TABLE VI
AVERAGE P–N VALUES WITH INCREASING VALUES OF THE STAGNATION TOLERANCE (Q) AT D = 10, 30, 50, AND 100

D \ Q	1	2	4	8	16	32	64	128	256	512
10	–8.125	–7.625	–7.375	–4.25	–1.375	5.625	11.625	**13.375**	10.75	7.25
30	9.75	10.5	12	13.25	16.375	**17.125**	16.75	15.875	13.125	7.625
50	13.25	14	15	16.25	17.625	**18.375**	17.875	16.625	13	6.875
100	11.25	12.5	11.875	**12.875**	**12.875**	**12.875**	11.75	9.25	4	1

$Q = 64$, 32, 32, 128, 32, 32, 16, and 64, for the considered algorithms: DE/best/1/bin, DE/rand/1/bin, DCMA-EA, DEGL, JADE, RBDE, SaDE, and SHADE, respectively. To summarize the results of P–N values on all of the considered DEs, the highest average P–N value indicates that the best choice of the stagnation tolerance is $Q = 32$ over CEC 2014 benchmark functions at $D = 30$ on average.

C. Scalability Analysis

The dimension of the test functions governs the difficulties on finding the global optimum. Higher dimensional functions are generally more difficult to solve. To understand the relationship between the dimensionality and the performance of the proposed framework, we evaluate the average P–N values of eight DE algorithms in CEC 2014 benchmark set with a stagnation tolerance (Q) varying from 1 to 512 at $D = 10$, 30, 50, and 100. In this way, the most suitable stagnation tolerance (Q) can be derived for each dimension.

Table VI shows the results of the considered DE algorithms with a varying stagnation tolerance (Q) over the test functions. For $D = 10$, 30, and 50, the results are obtained from 51 independent runs. For $D = 100$, we approximate the results from only four independent runs. The best choices of the stagnation tolerance are $Q = 128$, 32, and 32 for $D = 10$, 30, and 50, respectively. For $D = 100$, the best values of the stagnation tolerance are $Q = 8$, 16, and 32 over the approximated P–N values. Overall, high stagnation tolerance (Q) provides the proposed SPS framework superior performance in lowly dimensional test functions. When the optimization problem is of many variables and hard to locate the global optimum, the incorporated DEs generally perform well with low stagnation tolerance (Q).

D. Overall Performance

The demonstration of the experimental results is concluded by providing an overall performance comparison over all the benchmark functions, specifically, four real-world optimization problems and totally 90 test functions in CEC 2014 benchmark set at $D = 10$, 30, and 50. To show the overall performance, we draw the empirical cumulative probability distribution function (ECDF) of the normalized solution errors (NSE).

The NSE measure tries to derive the relative performance of an algorithm by adjusting the solution errors (SE) to a notionally common scale, specifically, 0 to 1. The NSE measure of an algorithm A on an objective function f in the rth independent run ($\mathrm{NSE}_{A,f,r}$) is computed by the unity-based normalization, more specifically

$$\mathrm{NSE}_{A,f,r} = \frac{\mathrm{SE}_{A,f,r} - \min\limits_{A,r}\left(\mathrm{SE}_{A,f,r}\right) + \varepsilon}{\max\limits_{A,r}\left(\mathrm{SE}_{A,f,r}\right) - \min\limits_{A,r}\left(\mathrm{SE}_{A,f,r}\right) + \varepsilon} \tag{20}$$

where $0 < \varepsilon \ll 1$ is a small real constant number to avoid zero values in the numerator and denominator. In the NSE, the value NSE = 1 correspond to the worst performance in certain optimization problem.

ECDF of the NSEs is a cumulative distribution function of the NSE, which is defined as

$$\mathrm{ECDF}(x) = \frac{1}{n_A \times n_f \times n_r} \sum_{i=1}^{n_A} \sum_{j=1}^{n_f} \sum_{k=1}^{n_r} I\left(\mathrm{NSE}_{i,j,k} \le x\right) \tag{21}$$

where n_A is the number of algorithms, n_f is the number of objective functions, n_r is the number of independent runs, and $I(\cdot)$ is an indicator function. In (21), the ECDF evaluates the empirical probability by seeing the numbers of NSEs that are smaller or equal to x. Therefore, larger values of ECDF for the same argument express better performance.

First, we compute the NSEs for eight considered algorithms and their SPSvariants in 94 objective functions with 51 independent runs. Then, the NSEs are separated into two sets: 1) those of the original DE algorithms and 2) those of the proposed SPS-DEs. Finally, we compute the ECDF for each set. This allows an overall comparison between the original DE algorithms and the incorporated DE algorithms.

Fig. 5 shows the ECDF of NSEs for all the original DE algorithms versus their SPS variants with the stagnation tolerance $Q = 32$ for four real-world optimization problems and 90 test functions in CEC 2014 at $D = 10$, 30, and 50. The ECDF curve of the DEs with the proposed SPS framework is almost always above that of the original DEs. The considered DE algorithms incorporated with the proposed method significantly outperform the corresponding original DE algorithms in most cases.

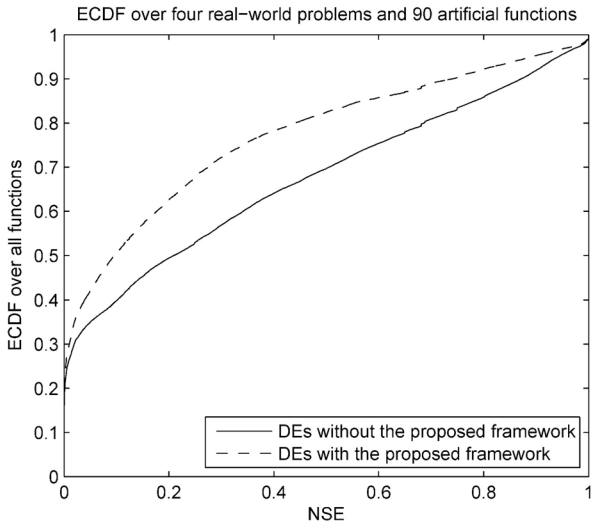


Fig. 5. Empirical cumulative probability distribution of normalized solution error of all DE algorithms against the corresponding SPS variants over four real-world optimization problems in CEC 2011 and 90 test functions in CEC 2014 benchmark functions at $D = 10$, 30, and 50.

Accordingly, the proposed SPS framework exhibits convincing improvement in the considered real-world optimization problems and benchmark functions on average.

VII. Conclusion

In this paper, an effective and efficient SPS framework is proposed to improve the performance of DE algorithms by providing alternative for the selection of parents during mutation and crossover. The proposed method adapts the selection of parents by storing successful solutions into an archive, and then the parents are selected from the archive when stagnation is occurred. The stagnation is detected by counting for the consecutive unsuccessful updates. If a solution is continuously not updated for more than a stagnation tolerance Q, the proposed method will switch the source of parents to the stored successful solutions. The operations of the proposed framework are very simple, so that the overall time complexity of the incorporated DE algorithms is not considerably increased. To balance the exploration and exploitation power of the proposed framework, the best choice of the stagnation tolerance $Q = 32$ is optimized for the considered eight DE algorithms in 30 and 50-D CEC 2014 benchmark functions.

The simulation results show that the proposed framework significantly improves performance of the considered DE algorithms in terms of solution error measure. In addition, the incorporation of the proposed SPS framework in eight DE algorithms demonstrates faster convergence speed on the distance between solutions and their centroid. Moreover, the incorporated DE algorithms generate fewer unsuccessful updates. To conclude, the proposed framework provides more promising solutions to guide the evolution and effectively helps DE algorithms escaping the situation of stagnation.

References

[1] R. Storn and K. V. Price, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," Int. Comput. Sci. Inst., Berkeley, CA, USA, Tech. Rep. TR-95-012, 1995.

[2] R. Storn and K. V. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[3] R. Storn, K. V. Price, and J. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization*. Berlin, Germany: Springer, 2005.

[4] R. Storn and K. V. Price, "Minimizing the real functions of the ICEC 1996 contest by differential evolution," in *Proc. IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, May 1996, pp. 842–844.

[5] 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 CEC'2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, Dec. 2009.

[6] 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 Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[7] P. Rocca, G. Oliveri, and A. Massa, "Differential evolution as applied to electromagnetics," *IEEE Antennas Propag. Mag.*, vol. 53, no. 1, pp. 38–49, Feb. 2011.

[8] C. C. Chiu, C. H. Sun, C. L. Li, and C. H. Huang, "Comparative study of some population-based optimization algorithms on inverse scattering of a two-dimensional perfectly conducting cylinder in dielectric slab medium," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 4, pp. 2302–2315, Apr. 2013.

[9] B. Liu *et al.*, "An efficient method for antenna design optimization based on evolutionary computation and machine learning techniques," *IEEE Trans. Antennas Propag.*, vol. 62, no. 1, pp. 7–18, Jan. 2014.

[10] U. Maulik and I. Saha, "Automatic fuzzy clustering using modified differential evolution for image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 9, pp. 3503–3510, Sep. 2010.

[11] I. Saha, U. Maulik, S. Bandyopadhyay, and D. Plewczynski, "SVMeFC: SVM ensemble fuzzy clustering for satellite image segmentation," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 1, pp. 52–55, Jan. 2012.

[12] S. Sarkar and S. Das, "Multilevel image thresholding based on 2D histogram and maximum Tsallis entropy—A differential evolution approach," *IEEE Trans. Image Process*., vol. 22, no. 12, pp. 4788–4797, Dec. 2013.

[13] G. W. Greenwood, "Using differential evolution for a subclass of graph theory problems," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1190–1192, Oct. 2009.

[14] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[15] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[17] J. Zhang and A. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

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

[19] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 482–500, Apr. 2012.

[20] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, México, Jun. 2013, pp. 71–78.

[21] R. Tanabe and A. Fukunaga, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, México, Jun. 2013, pp. 1952–1959.

[22] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.

[23] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Dept. Electron. Telecommun. Eng., Jadavpur Univ., Kolkata, India, and School Electr. Electron. Eng., Nanyang Technol. Univ., Singapore, Dec. 2010.

[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," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, and Nanyang Technol. Univ., Singapore, Tech. Rep. 201311, Dec. 2013.

[25] M. Vasile, E. Minisci, and M. Locatelli, "An inflationary differential evolution algorithm for space trajectory optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 2, pp. 267–281, Apr. 2011.

[26] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[27] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Comput. Fusion Found. Methodol. Appl.*, vol. 15, no. 11, pp. 2127–2140, Nov. 2011.

[28] H. Wang, S. Rahnamayan, and S. Zeng, "Generalised opposition-based differential evolution: An experimental study," *Int. J. Comput. Appl. Technol.*, vol. 43, no. 4, pp. 311–319, Jan. 2012.

[29] S. Ghosh, S. Das, S. Roy, S. K. Minhazul, and P. N. Suganthan, "A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization," *Inf. Sci.*, vol. 182, no. 1, pp. 199–219, Jan. 2012.

[30] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

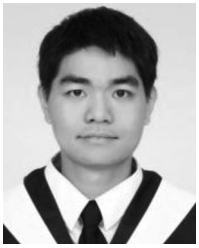
[31] N. Hansen, A. S. P. Niederberger, L. Guzzella, and P. Koumoutsakos, "A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 180–197, Feb. 2009.
[32] Y. Zhou, X. Li, and L. Gao, "A novel two-layer hierarchical differential evolution algorithm for global optimization," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Manchester, U.K., Oct. 2013, pp. 2916–2921.
[33] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 634–647, Apr. 2013.
[34] Y. Wang, H. X. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," *Appl. Soft Comput.*, vol. 18, pp. 232–247, May 2014.
[35] S. M. Guo and C. C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 31–49, Feb. 2015.
[36] Y. Wang, Z. X. Cai, and Q. F. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Inf. Sci.*, vol. 185, no. 1, pp. 153–177, Feb. 2012.
[37] B. Dorronsoro and P. Bouvry, "Improving classical and decentralized differential evolution with new mutation operator and population topologies," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 67–98, Feb. 2011.
[38] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.
[39] S. Elsayed, R. Sarker, and D. Essam, "Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, New Orleans, LA, USA, Jun. 2011, pp. 1041–1048.
[40] J. Brest and M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Comput. Fusion Found. Methodol. Appl.*, vol. 15, no. 11, pp. 2157–2174, Nov. 2011.
[41] Q. K. Pan, P. N. Suganthan, L. Wang, L. Gao, and R. Mallipeddi, "A differential evolution algorithm with self-adaptive strategy and control parameters," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 394–408, Jan. 2011.
[42] W. Y. Gong, Z. H. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 397–413, Apr. 2011.
[43] S. Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large-scale optimization," *Soft Comput. Fusion Found. Methodol. Appl.*, vol. 15, no. 11, pp. 2175–2185, Nov. 2011.
[44] W. Gong, Á. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: An empirical study," *Inf. Sci.*, vol. 181, no. 24, pp. 5364–5386, Dec. 2011.
[45] W. Gong, Z. Cai, and Y. Wang, "Repairing the crossover rate in adaptive differential evolution," *Appl. Soft Comput.*, vol. 15, pp. 149–168, Feb. 2014.
[46] J. Aalto and J. Lampinen, "A mutation adaptation mechanism for differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, México, Jun. 2013, pp. 55–62.
[47] M. Weber, V. Tirronen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Comput. Fusion Found. Methodol. Appl.*, vol. 14, no. 11, pp. 1187–1207, Sep. 2010.
[48] S. M. Elsayed, R. A. Sarker, and T. Ray, "Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, México, Jun. 2013, pp. 1932–1937.
[49] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 689–707, Oct. 2014.
[50] A. Ghosh, S. Das, A. Chowdhury, and R. Giri, "An improved differential evolution algorithm with fitness-based adaptation of the control parameters," *Inf. Sci.*, vol. 181, no. 18, pp. 3749–3765, Sep. 2011.
[51] R. Li, L. Xu, X. W. Shi, N. Zhang, and Z. Q. Lv, "Improved differential evolution strategy for antenna array pattern synthesis problems," *Prog. Electromagn. Res.*, vol. 113, pp. 429–441, Feb. 2011.
[52] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2202–2215, Dec. 2013.
[53] W. Y. Gong and Z. H. Cai, "Differential evolution with ranking-based mutation operators," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2066–2081, Dec. 2013.
[54] Y. Lou, J. Li, and G. Li, "A differential evolution algorithm based on individual-sorting and individual-sampling strategies," *J. Comput. Inf. Syst.*, vol. 8, no. 2, pp. 717–725, 2012.
[55] A. M. Sutton, M. Lunacek, and L. D. Whitley, "Differential evolution and non-separability: Using selective pressure to focus search," in *Proc. 9th Annu. Conf. Genet. Evol. Comput. (GECCO)*, London, U.K., Jul. 2007, pp. 1428–1435.
[56] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proc. 6th Int. Mendel Conf. Soft Comput.*, Brno, Czech Republic, Jun. 2000, pp. 76–83.
[57] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adaptive control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.
[58] J. Cheng, G. Zhang, and F. Neri, "Enhancing distributed differential evolution with multicultural migration for global numerical optimization," *Inf. Sci.*, vol. 247, no. 20, pp. 72–93, Oct. 2013.
[59] A. Zamuda, J. Brest, and E. Mezura-Montes, "Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, México, Jun. 2013, pp. 1925–1931.
[60] J. Tvrdík, "Adaptation in differential evolution: A numerical comparison," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1149–1155, 2009.
[61] J. Tvrdík, R. Poláková, J. Veselský, and P. Bujok, "Adaptive variants of differential evolution: Towards control-parameter-free optimizers," in *Handbook of Optimization* (Intelligent Systems Reference Library), vol. 38, I. Zelinka, A. Abraham, and V. Snasel, Eds. London, U.K.: Springer, 2013, pp. 423–449.
[62] J. Brest *et al.*, "Differential evolution and differential ant-stigmergy on dynamic optimisation problems," *Int. J. Syst. Sci.*, vol. 44, no. 4, pp. 663–679, 2013.

Shu-Mei Guo (M'11) received the M.S. degree from the Department of Computer and Information Science, New Jersey Institute of Technology, Newark, NJ, USA, and the Ph.D. degree in computer and systems engineering from University of Houston, Houston, TX, USA, in 1987 and 2000, respectively.

Since 2000 she has been an Assistant Professor with the Department of Computer System and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, and since 2010 she has been a Full Professor. Her research interests include various applications on evolutionary programming, image processing, chaos systems, Kalman filtering, fuzzy methodology, sampled-data systems, and computer and systems engineering.

Chin-Chang Yang (S'14) was born in Taipei, Taiwan, in 1985. He received the M.S. degree from the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, in 2009.

His research interests include evolutionary computation, fuzzy system, and its applications to image processing.



Pang-Han Hsu (S'14) was born in Hsinchu, Taiwan, in 1991. He received the B.S. degree from the Department of Information Management, National Central University, Zhongli, Taiwan, in 2013.

His research interests include evolutionary computation and image filtering.

Jason S.-H. Tsai received the M.S. and Ph.D. degrees in electrical engineering from University of Houston, Houston, TX, USA, in 1985 and 1988, respectively.

In 1988 he was an Associate Professor with the Department of Electrical Engineering, National Cheng-Kung University, Tainan, Taiwan, where he became a Full Professor in 1992, and has also been a Distinguished Professor since 2002. His research interests include state–space self-tuning control, chaotic system control, partial differential system control, numerical analysis, and robotics.

Prof. Tsai was an Editor of *Journal of the Chinese Institute of Electrical Engineering* from 1997 to 2002. He has been the (Executive) Editor of *Science Development* (National Science Council) since 2001 and has been an Associate Editor of *International Journal of Systems Science* since 2003 and *Journal of the Franklin Institute* since 2007.