Reviewing and Benchmarking Parameter Control Methods in Differential Evolution

Ryoji Tanabe, *Member, IEEE*, and Alex Fukunaga

***Abstract*—Many differential evolution (DE) algorithms with various parameter control methods (PCMs) have been proposed. However, previous studies usually considered PCMs to be an integral component of a complex DE algorithm. Thus, the characteristics and performance of each method are poorly understood. We present an in-depth review of 24 PCMs for the scale factor and crossover rate in DE and a large-scale benchmarking study. We carefully extract the 24 PCMs from their original, complex algorithms and describe them according to a systematic manner. Our review facilitates the understanding of similarities and differences between existing, representative PCMs. The performance of DEs with the 24 PCMs and 16 variation operators is investigated on 24 black-box benchmark functions. Our benchmarking results reveal which methods exhibit high performance when embedded in a standardized framework under 16 different conditions, independent from their original, complex algorithms. We also investigate how much room there is for further improvement of PCMs by comparing the 24 methods with an oracle-based model, which can be considered to be a conservative lower bound on the performance of an optimal method.**

***Index Terms*—Benchmarking study, continuous optimization, differential evolution (DE), parameter control methods (PCMs), review.**

I. Introduction

THE PERFORMANCE of evolutionary algorithms (EAs) is significantly influenced by the control parameter settings [1], [2]. The optimal control parameter settings depend on the characteristics of the problem instance which is being solved, and it is necessary to tune the control parameters to obtain the good performance of an EA for a given problem instance. Suitable control parameter settings are also dependent on the state of the search progress of an EA. Thus, parameter control methods (PCMs), which are the methods for automatically tuning the control parameters during the search process, have been widely studied [1], [2]. According to [1], PCMs can be classified into the following three broad categories.

1) Deterministic PCMs (DPCMs) which set the control parameter values according to a deterministic rule without using any information obtained during the search process. DPCMs include, for example, a method that decreases mutation rates according to the computational resources used so far [the number of function evaluations (FEvals) or runtime] [3].
2) Adaptive PCMs (APCMs) which set the global control parameters according to the current state of the search. To our knowledge, the earliest APCM is the "1/5 success rule" for evolution strategy (ES), which sets the step size parameter σ according to a simple adaptive criterion [4].
3) Self-adaptive Parameter Control Methods (SPCMs), in which each individual in a population has its own associated control parameter values. These individual-specific control parameter values are encoded as part of the individual and, thus, the control parameters themselves are subject to selection as well as modification by genetic operators, with the aim of simultaneously evolving the individuals as well as the individual-specific control parameters. Representative instances of this approach include Self-adaptive ES [5]. The key difference between APCMs and SPCMs is whether new control parameter values are generated by applying variation operators to values associated with each individual.

These individual-specific control parameter values are encoded as part of the individual and, thus, the control parameters themselves are subject to selection as well as modification by genetic operators, with the aim of simultaneously evolving the individuals as well as the individual-specific control parameters. Representative instances of this approach include self-adaptive ES [5]. The key difference between APCMs and SPCMs is whether new control parameter values are generated by applying variation operators to values associated with each individual.

Differential evolution (DE) is an EA that was primarily designed for continuous optimization problems [6]. Continuous optimization is the problem of finding a real-valued vector $\boldsymbol{x} = (x_1, \ldots, x_D)^T \in \mathbb{S} \subseteq \mathbb{R}^D$ that minimizes

Manuscript received January 22, 2018; revised April 26, 2018, July 21, 2018, and November 5, 2018; accepted January 9, 2019. This work was supported in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X386, in part by the Shenzhen Peacock Plan under Grant KQTD2016112514355531, in part by the Science and Technology Innovation Committee Foundation of Shenzhen under Grant ZDSYS201703031748284, and in part by the Program for University Key Laboratory of Guangdong Province under Grant 2017KSYS008. This paper was recommended by Associate Editor S. Yang. *(Corresponding author: Ryoji Tanabe.)*

R. Tanabe is with the Shenzhen Key Laboratory of Computational Intelligence, University Key Laboratory of Evolving Intelligent Systems, Guangdong Province, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: rt.ryoji.tanabe@gmail.com).

A. Fukunaga is with the Graduate School of Arts and Sciences, University of Tokyo, Tokyo 151-0063, Japan (e-mail: fukunaga@idea.c.u-tokyo.ac.jp).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

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/TCYB.2019.2892735

an objective function $f : \mathbb{S} \rightarrow \mathbb{R}$, where D is the dimensionality of the problem and $\mathbb{S}$ is the feasible region of the search space. Despite its relative simplicity, DE has been shown to be competitive with more complex optimization algorithms and has been applied to many practical problems [7]–[9].

In the evolutionary computation community, it was widely accepted for decades that the performance of EAs was strongly influenced by their control parameter settings. Nevertheless, it was initially reported that the performance of DE was fairly robust with respect to control parameter settings [6]. However, later research showed that in fact, the effectiveness of DE was significantly affected by control parameter settings [10]–[12]. The main control parameters of DE are the population size N, the scale parameter $F > 0$, and the crossover rate $C \in [0, 1]$. As a result, PCMs for DE have been an active area of research since around 2005. Many DE algorithms with PCMs have been proposed [8], [9], [13], [14].

In general, the term "a DE algorithm with a PCM" denotes a complex algorithm composed of multiple components. For example, "L-SHADE" [15] consists of four key components: 1) current-to-*p*best/1 mutation strategy [16]; 2) binomial crossover; 3) "SHADE method" for adapting parameters F and C [i.e., PCM of success-history-based adaptive DE (PCM-SHADE)]; and 4) linear population size reduction strategy. In the same manner, "CoDE" [17] consists of four components: 1) three mutation strategies (rand/1, rand/2, and current-to-rand/1); 2) binomial crossover; 3) the "CoDE method" for assigning F and C values to each individual (i.e., PCM-CoDE); and 4) simultaneous generation of three children per an individual.

While there have been a number of performance comparisons among such complex DE algorithms, recent work [18], [19] pointed out that there are few comparative studies of PCMs *in isolation*. For example, a previous study [15] compares L-SHADE with five complex DE algorithms with PCMs (SaDE [20], dynNP-jDE [21], JADE [16], EPSDE [22], and CoDE [17]), but a performance comparison between their PCMs [PCM-SHADE, PCM of SaDE (PCM-SaDE), PCM of Janez's DE (PCM-jDE), PCM of Jingqiao and Arthur's DE (PCM-JADE), PCM of EPSDE (PCM-EPSDE), and PCM-CoDE] has never been performed in the literature. The results in [15] show that L-SHADE performs better than the five DE algorithms. However, the results do not mean that the performance of PCM-SHADE is better than that of the five PCMs, because it is unclear which component [1)–4) in the above example] makes the largest contribution to the better performance of L-SHADE. Thus, the performance of 3) PCMs is currently poorly understood—a deeper understanding is necessary to enable the development of new PCMs and further advance the state-of-the-art.

In this paper, we first present an *in-depth* review of 24 PCMs in DE which yields novel insights. Existing survey papers (e.g., [2], [8], [9], [13], [14], and [23]) have a much broader scope, covering all aspects of DE. *In contrast, this paper focuses only on PCMs for the scale factor F and crossover rate C*. We extracted only 3) PCMs from complex DE algorithms and generalized them so that they can be combined with arbitrary mutation strategies and crossover methods. Instead of an exhaustive, shallow survey of PCMs, we precisely describe the 24 PCMs and clarify their relationship to each other.

Next, we present a large-scale benchmarking study which investigates the performance of the 24 PCMs on the noiseless black-box optimization benchmarking (BBOB) benchmark set [24]. Each PCM is incorporated into DE algorithms with eight mutation strategies and two crossover methods. By fixing the 1) mutation strategy and 2) crossover method for all PCMs, the performance of 3) PCMs can be investigated independently. Furthermore, we evaluate the performance of the 24 PCMs with hyperparameters tuned by SMAC [25], which is an automatic algorithm configurator.

Finally, the 24 PCMs are compared with GAODE [26], a recently proposed, oracle-based model of an "ideal PCM," which can be considered as a lower bound on the performance of PCMs in DE. This shows that the performance of existing PCMs is poorly compared with an (approximately) ideal PCM and indicates that there is a significant room for improvement and opportunities for future work in the development of PCMs.

This paper is the first study, which reviews many PCMs for DE and performs a systematic investigation of their performance in isolation. While there have been some benchmarking studies of DE algorithms with PCMs (e.g., [27] and [28]) as well as survey papers [8], [9], [13], [14], and [23], they did not focus specifically on PCMs. Several previous studies have compared the performance of PCMs (e.g., [18], [19], [29], and [30]). However, the benchmark problems used in the previous studies [18], [19] were constrained continuous optimization and multiobjective optimization problems. Thus, the performance of PCMs on bound-constrained continuous optimization problems, the most basic class of optimization problems as represented by the BBOB benchmarks, has not been thoroughly investigated. Also, the previous studies [29], [30] investigate the adaptation mechanisms only for either the scale factor or crossover rate, but not both. Most importantly, all of the previous works [18], [19], [29], [30] evaluated the performance of only a few PCMs, and only considered up to two combinations of mutation and crossover methods.

In most previous studies, PCMs have been considered integral, algorithmic components of complex DE algorithms. Thus, the similarities and differences between existing PCMs were often not obvious. Our review classifies and organizes existing PCMs by presenting a detailed, precise description of each method in a common framework. This should facilitate the understanding of the characteristics of each PCM as well as the relationship among the various methods. Furthermore, there has a recent tendency for "horse-race" algorithm development in the DE community—many DE algorithms that include new PCMs embedded in a complex DE have been proposed, where the performance of the entire algorithm is evaluated without isolating the performance impact of the PCMs. Our benchmarking study aims to reveal which PCM tend to perform when embedded in a standard DE framework, rather than which complex DE algorithm has a good performance. Source code for the 24 PCMs as well as all experimental data presented in this paper are available online (https://sites.google.com/view/pcmde/) so that the researchers

can easily compare their newly developed PCMs with the 24 PCMs. We hope that our review and benchmarking results encourage further development of PCMs for DE.

This paper is organized as follows. First, Section II describes the basic DE framework. Then, Section III reviews the PCMs proposed in the literature. The historical background of PCMs, their characteristics, and the relationships between different PCMs are discussed in Section IV. The benchmarking results are reported in Section V. Finally, Section VI concludes this paper and discusses directions for future work.

II. Differential Evolution

This section briefly describes the basic DE algorithm [6] without any PCM (i.e., the F and C parameters are fixed values, such as $F = 0.5$ and $C = 0.9$). A DE population $\boldsymbol{P} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\}$ is represented as a set of real parameter vectors $\boldsymbol{x}^i = (x^i_1, \ldots, x^i_D)^T$, $i \in \{1, \ldots, N\}$, where D is the dimensionality of a given problem and N is the population size. The detailed pseudocode for the DE described here can be found in Algorithm S.1 in the supplementary file.

After the initialization of the population, for each iteration t, for each $\boldsymbol{x}^{i,t}$, a mutant vector $\boldsymbol{v}^{i,t}$ is generated from the individuals in $\boldsymbol{P}^t$ by applying a mutation strategy. Table S.1 in the supplementary file shows eight representative mutation strategies for DE (rand/1, rand/2, best/1, best/2, current-to-rand/1, current-to-best/1 [16], current-to-$p$best/1, and rand-to-$p$best/1 [31]). The scale factor F in Table S.1 in the supplementary file controls the magnitude of the differential mutation.

Then, the mutant vector $\boldsymbol{v}^{i,t}$ is crossed with the parent $\boldsymbol{x}^{i,t}$ to generate a trial vector $\boldsymbol{u}^{i,t}$. Binomial crossover (bin) and exponential crossover (exp) are most commonly used in DE. However, the performance of a DE algorithm using exponential crossover significantly depends on the ordering of variable indices [32]. Shuffled exponential crossover (sec), which applies exponential crossover after shuffling the variable indices of parent individuals, addresses this issue [7], [32]. Algorithms S.2–S.4 in the supplementary file show the three crossover methods, respectively. The crossover rate C in Algorithms S.2–S.4 in the supplementary file controls the number of inherited variables from $\boldsymbol{x}^{i,t}$ to $\boldsymbol{u}^{i,t}$.

After all of the trial vectors have been generated, for each i, $\boldsymbol{x}^{i,t}$ is compared with its corresponding $\boldsymbol{u}^{i,t}$, keeping the better individual in the population $\boldsymbol{P}^{t+1}$. The parent individuals that were worse than the trial vectors (and are therefore not selected for survival in the standard DE) are preserved in an external archive $\boldsymbol{A}$. Whenever the size of the archive exceeds $|\boldsymbol{A}|$, randomly selected individuals are deleted to keep the archive size constant. Individuals in $\boldsymbol{A}$ are used for the current-to-$p$best/1 and rand-to-$p$best/1 mutation strategies.

III. Parameter Control Methods in DE

As mentioned in Section I, PCMs can be classified into the following three categories: 1) DPCMs; 2) APCMs; and 3) SPCMs. This section reviews 24 PCMs for DE in each category in Sections III-A–III-C, respectively. The properties of the 24 PCMs are summarized in Table I, which is explained in Section IV. For the sake of clarity, MDE_pBX [33], MDE [34], and ADE [35] are denoted as IMDE, ZMDE, and YADE, respectively. To improve understandability, we describe PCM-jDE [12], PCM of fitness diversity self-adaptive DE (PCM-FDSADE) [36], and PCM of improved self-adaptive DE (PCM-ISADE) [37] in a revised manner from their original description. Some papers provide insufficient details on the proposed PCMs to allow reimplementation, so we excluded such insufficiently specified PCMs from this paper.

TABLE I
Properties of 24 PCMs for DE. Each Symbol Is Explained in Section IV-B

	F	C	S/O	C/D	S/M	Info.	Inh.
PCM-DERSF [38]	DPCM	0.9		C	M	N	
PCM-DETVSF [38]	DPCM	0.9		C	S	T	
PCM-CoDE [17]	DPCM	DPCM		D	M	N	
PCM-ZMDE [34]	DPCM	DPCM		C	M	N	
PCM-SinDE [39]	DPCM	DPCM		C	S	T	
PCM-SWDE [40]	DPCM	DPCM		D	M	N	
PCM-DEPD [41]	APCM	0.5	O	C	S	P	
PCM-jDE [12]	APCM	APCM	S	C	M	N	✓
PCM-cDE [42]	APCM	APCM	S	D	M	H	
PCM-SaNSDE [44]	APCM	APCM	S	C	M	H	
PCM-FDSADE [36]	APCM	APCM	S&O	C	M	P	✓
PCM-ISADE [37]	APCM	APCM	S&O	C	M	P	✓
PCM-SaDE [20]	DPCM	APCM	S	C	M	H	
PCM-JADE [16]	APCM	APCM	S	C	M	H	
PCM-EPSDE [22]	APCM	APCM	S	D	M	N	✓
PCM-RDE [51]	APCM	APCM	O	C	M	P	
PCM-IMDE [33]	APCM	APCM	S	C	M	H	
PCM-SHADE [46]	APCM	APCM	S	C	M	H	
PCM-YADE [35]	APCM	APCM	O	C	M	P	
PCM-DEDPS [50]	APCM	APCM	S	D	M	H	
PCM-CoBiDE [49]	APCM	APCM	S	C	M	N	✓
PCM-IDE [52]	APCM	APCM	O	C	M	P	
PCM-SLADE [48]	APCM	APCM	S	C	M	H	
PCM-SDE [53]	SPCM	0.5	S	C	M	P	✓

Zhang and Sanderson [16] pointed out that some APCMs are incorrectly classified by the original authors as SPCMs with respect to the taxonomy proposed in [1]. For example, self-adaptive DE (SaDE) [20] does not use any variation operators to generate the values of C. Thus, even though the name "SaDE" includes the term "self-adaptive," the PCM of SaDE should be classified as an APCM. Following Zhang and Sanderson's [16] guideline, we have classified PCMs according to their functional mechanisms, rather than their original names.

Algorithms S.5–S.28 in the supplementary file show a basic DE algorithm (see Section II) with each generalized PCM, respectively. Note that the pseudocode in Algorithms S.5–S.28 in the supplementary file describes the minimally modified DE algorithms to incorporate into each PCM but does not represent the exact DE algorithms described in the original paper. It should be re-emphasized that in this paper, we survey and evaluate PCMs, which are the parameter control components used in the previous DE algorithms (e.g., PCM-DERSF [38]), and not the complex DE algorithms that consist of multiple components (e.g., DERSF). See the original papers for full descriptions of the entire, complex, original algorithms which incorporates the 24 PCMs.

On the one hand, the $F_{i,t}$ and $C_{i,t}$ represent values of F and C assigned to an individual $\boldsymbol{x}_{i,t}$ ($i \in \{1, \ldots, N\}$) in the population $\boldsymbol{P}^t$ at iteration t (i.e., each individual uses different values of F and C). On the other hand, F_t and C_t represent values of F and C assigned to all individuals in $\boldsymbol{P}^t$ (i.e., all individuals use same values of F and C). The function randu$[a, b]$ denotes a uniformly selected random number from $[a, b]$, and randn(μ, σ^2) is a value selected randomly from a normal distribution with mean μ and variance σ^2. The function randc(μ, σ) selects values randomly from a Cauchy distribution with location parameter μ and scale parameter σ. Unless explicitly noted, when values of F and C generated by a PCM are outside of $[0, 1]$, they are replaced by the closest, limit value (0 or 1). We say that a generation of a trial vector is *successful* if $f(\boldsymbol{u}^{i,t}) \leq f(\boldsymbol{x}^{i,t})$. Otherwise, we say that the trial vector generation is a *failure*.

A. DPCMs in DE

1) PCM-DERSF (Algorithm S.5): The PCM of DE with random scale factor (PCM-DERSF) [38] uniformly randomly generates the value of $F_{i,t}$ in the range $[F^{\min}, F^{\max}]$ as follows: $F_{i,t} = \text{randu}[F^{\min}, F^{\max}]$. The recommended values of $F^{\min}$ and $F^{\max}$ are 0.5 and 1, respectively. It is possible to generate diversified mutant individuals by randomly generating the F parameter. While the F values are randomly generated, C is a constant value ($C = 0.9$ is recommended).

2) PCM-DETVSF (Algorithm S.6): In the PCM of DE with time varying scale factor (PCM-DETVSF) [38], the F value decreases linearly with the number of iterations t as follows: $F_t = (F^{\max} - F^{\min})([t^{\max} - t]/t^{\max}) + F^{\min}$, where $t^{\max}$ is the maximum number of iterations and F_1 and $F_{t^{\max}}$ are equal to $F^{\max}$ and $F^{\min}$, respectively. The recommended settings of $F^{\min}$ and $F^{\max}$ are 0.4 and 1.2, respectively. These settings are based on a general rule of thumb that the explorative and exploitative searches should be performed at the beginning and end of the search process. Larger and smaller F values are appropriate for each purpose. Similar to PCM-DERSF, C is a constant value ($C = 0.9$).

3) PCM-SinDE (Algorithm S.7): The PCM of sinusoidal DE (PCM-SinDE) [39] determines the values of F and C based on the sinusoidal function as follows:

$$F_t = \frac{1}{2}\left(\frac{t}{t^{\max}}(\sin(2\pi\omega t)) + 1\right) \tag{1}$$

$$C_t = \frac{1}{2}\left(\frac{t}{t^{\max}}(\sin(2\pi\omega t + \pi)) + 1\right) \tag{2}$$

where ω is the angular frequency, and a value of $\omega = 0.25$ is recommended. The amplitude of the F and C values depends on the number of iterations t. Thus, the ranges of the possible values of F and C increase as the search progresses.

4) PCM-ZMDE (Algorithm S.8): The PCM of Zou's modified DE (PCM-ZMDE) [34] randomly generates the F and C values according to the normal and uniform distributions as follows: $F_{i,t} = \text{randn}(0.75, 0.1)$ and $C_{i,t} = \text{randu}[0.8, 1]$.

5) PCM-CoDE (Algorithm S.9): The PCM of composite DE (PCM-CoDE) [17] uses three predefined pairs of F and C values for parameter control: $\boldsymbol{q}^1 = (1, 0.1)$, $\boldsymbol{q}^2 = (1, 0.9)$, and $\boldsymbol{q}^3 = (0.8, 0.2)$, where $\boldsymbol{q} = (F, C)$. These combinations were determined based on frequently used parameter settings in the DE community. At the beginning of each iteration t, a randomly selected pair $\boldsymbol{q}$ is assigned to each individual.

6) PCM-SWDE (Algorithm S.10): At the beginning of each iteration t, the PCM of switching DE (PCM-SWDE) [40] randomly assigns the extreme values of F (0.5 or 2) and C (0 or 1) to each individual. The small and large F values are helpful for the explorative and the exploitative searches, respectively. The use of the two extreme C values is to strike a balance between variable-wise and vector-wise searches.

B. APCMs in DE

1) PCM-DEPD (Algorithm S.11): In the PCM of DE using precalculated differential (PCM-DEPD) [41], only the F parameter is adaptively adjusted based on the objective values of individuals in the population $\boldsymbol{P}^t$. For each iteration t, F_t is given as follows:

$$F_t = \begin{cases} \max\left\{F^{\min}, 1 - |\frac{f_t^{\max}}{f_t^{\min}}|\right\} & \text{if } |\frac{f_t^{\max}}{f_t^{\min}}| < 1 \\ \max\left\{F^{\min}, 1 - |\frac{f_t^{\min}}{f_t^{\max}}|\right\} & \text{otherwise} \end{cases} \tag{3}$$

where $f_t^{\max}$ and $f_t^{\min}$ are the maximum and minimum objective values in $\boldsymbol{P}^t$. The recommended value for $F^{\min}$ is 0.4. Unlike F, C is a fixed value ($C = 0.5$).

2) PCM-jDE (Algorithm S.12): The PCM-jDE [12] assigns a different set of parameter values $F_{i,t}$ and $C_{i,t}$ to each $\boldsymbol{x}^{i,t}$ in $\boldsymbol{P}^t$. For $t = 1$, the parameters for all individuals $\boldsymbol{x}^{i,t}$ are set to $F_{i,t} = 0.5$ and $C_{i,t} = 0.9$. In each iteration t, $F_{i,t}^{\text{trial}}$ and $C_{i,t}^{\text{trial}}$ are generated as follows:

$$F_{i,t}^{\text{trial}} = \begin{cases} \text{randu}[0.1, 1] & \text{if randu}[0, 1] < \tau_F \\ F_{i,t} & \text{otherwise} \end{cases} \tag{4}$$

$$C_{i,t}^{\text{trial}} = \begin{cases} \text{randu}[0, 1] & \text{if randu}[0, 1] < \tau_C \\ C_{i,t} & \text{otherwise} \end{cases} \tag{5}$$

where τ_F and $\tau_C \in (0, 1]$ are the hyperparameters for parameter adaptation ($\tau_F = \tau_C = 0.1$ is the recommended setting). For each individual, $\boldsymbol{u}^{i,t}$ is generated by using $F_{i,t}^{\text{trial}}$ and $C_{i,t}^{\text{trial}}$ in (4) and (5), respectively. When the trial is a success, each individual $\boldsymbol{x}^{i,t}$ uses $F_{i,t}^{\text{trial}}$ and $C_{i,t}^{\text{trial}}$ in the next iteration (i.e., $F_{i,t+1} = F_{i,t}^{\text{trial}}$ and $C_{i,t+1} = C_{i,t}^{\text{trial}}$). Otherwise, each individual keeps using $F_{i,t}$ and $C_{i,t}$ (i.e., $F_{i,t+1} = F_{i,t}$ and $C_{i,t+1} = C_{i,t}$).

3) PCM-FDSADE (Algorithm S.13): The PCM-FDSADE [36] is a variant of PCM-jDE. Similar to PCM-jDE, PCM-FDSADE generates $F_{i,t}^{\text{trial}}$ and $C_{i,t}^{\text{trial}}$ for each individual and updates $F_{i,t+1}$ and $C_{i,t+1}$ based on the success/failure decision. However, PCM-FDSADE samples $F_{i,t}^{\text{trial}}$ and $C_{i,t}^{\text{trial}}$ based on the diversity of the objective values of individuals as follows:

$$F_{i,t}^{\text{trial}} = \begin{cases} \text{randu}[0.1, 1] & \text{if randu}[0, 1] < K(1 - \phi_t) \\ F_{i,t} & \text{otherwise} \end{cases} \tag{6}$$

$$C_{i,t}^{\text{trial}} = \begin{cases} \text{randu}[0, 1] & \text{if randu}[0, 1] < K(1 - \phi_t) \\ C_{i,t} & \text{otherwise} \end{cases} \tag{7}$$

where K is the control parameter of PCM-FDSADE ($K = 0.3$ is recommended). The value of ϕ_t indicates the diversity of the population $\boldsymbol{P}^t$ in the objective space, where $\phi_t = f_t^{\text{std}}/(f_t^{\max} - f_t^{\min})$, f_t^{std} is the standard deviation of $f(\boldsymbol{x}^{1,t}), \ldots, f(\boldsymbol{x}^{N,t})$, and

$f_t^{\max}$ and $f_t^{\min}$ are their maximum and minimum objective values, respectively. If $f_t^{\max} - f_t^{\min} < 0$, $\phi_t = 0$.

4) PCM-ISADE (Algorithm S.14): The PCM-ISADE [37] is also based on PCM-jDE, but $F_{i,t}^{\text{trial}}$ and $C_{i,t}^{\text{trial}}$ are generated based on the objective values of individuals $f(\boldsymbol{x}^{1,t}), \ldots, f(\boldsymbol{x}^{N,t})$ as follows:

$$F_{i,t}^{\text{trial}} = \begin{cases} \alpha\big(F_{i,t} - 0.1\big) + 0.1 & \text{if randu}[0, 1] < \tau_F \,\&\, f\big(\boldsymbol{x}^{i,t}\big) < f_t^{\text{avg}} \\ \text{randu}[0.1, 1] & \text{if randu}[0, 1] < \tau_F \,\&\, f\big(\boldsymbol{x}^{i,t}\big) \geq f_t^{\text{avg}} \\ F_{i,t} & \text{otherwise} \end{cases} \tag{8}$$

$$C_{i,t}^{\text{trial}} = \begin{cases} \alpha\, C_{i,t} & \text{if randu}[0, 1] < \tau_C \,\&\, f\big(\boldsymbol{x}^{i,t}\big) < f_t^{\text{avg}} \\ \text{randu}[0, 1] & \text{if randu}[0, 1] < \tau_C \,\&\, f\big(\boldsymbol{x}^{i,t}\big) \geq f_t^{\text{avg}} \\ C_{i,t} & \text{otherwise} \end{cases} \tag{9}$$

where $\alpha = (f(\boldsymbol{x}^{i,t}) - f_t^{\min})/(f_t^{\text{avg}} - f_t^{\min})$ and f_t^{avg} is the average objective value of individuals in $\boldsymbol{P}^t$. The recommended values of τ_F and τ_C are 0.1.

5) PCM-cDE (Algorithm S.15): The PCM of competitive DE (PCM-cDE) [42] adaptively selects a combination of F and C values from a predefined parameter pool. Although several variants of PCM-cDE (e.g., [28]) have been proposed, we describe the original version in [42]. For F and C, $\boldsymbol{F}^{\text{pool}} = \{0.5, 0.8, 1\}$ and $\boldsymbol{C}^{\text{pool}} = \{0, 0.5, 1\}$ are defined. There are nine possible combinations of the F and C values from each pool, denoted as $\boldsymbol{q}^1 = (0.5, 0), \ldots, \boldsymbol{q}^9 = (1, 1)$.

For each t, a pair of parameters assigned for each individual $\boldsymbol{x}^{i,t}$ is selected from $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^9$. The selection probability $s_{k,t} \in (0, 1]$ of selecting $\boldsymbol{q}^k$ ($k \in \{1, \ldots, 9\}$) is given as follows:

$$s_{k,t} = \frac{n_k^{\text{succ}} + n^0}{\sum_{l=1}^{9}\big(n_l^{\text{succ}} + n^0\big)} \tag{10}$$

where n^0 is a parameter to avoid $s_{k,t} = 0$. In (10), n_k^{succ} represents the number of successful trials of $\boldsymbol{q}^k$ from the last initialization. When any $s_{k,t}$ is below the threshold δ, all of the n^{succ} values are reinitialized to 0. The recommended settings of n^0 and δ are 2 and 1/45, respectively.

6) PCM-SaDE (Algorithm S.16): PCM-SaDE [20], $F_{i,t}$ and $C_{i,t}$ are generated as follows: $F_{i,t} = \text{randn}(0.5, 0.3)$ and $C_{i,t} = \text{randn}(\mu_C, 0.1)$. Even when $F_{i,t}$ is outside of $[0, 1]$, a repair method is not applied. The mean μ_C is set to 0.5 at the beginning of the search. While the values of F are randomly generated, the C parameter is adaptively adjusted. For each iteration, successful C parameters are stored into a historical memory $\boldsymbol{H}^C$. When the number of iterations t exceeds a learning period t^{learn}, μ_C is set to the median value[1] of all elements in $\boldsymbol{H}^C$. For example, if elements in the historical memory are $\boldsymbol{H}^{C,1} = \{0.1, 0.2\}$, $\boldsymbol{H}^{C,2} = \{0.3\}$, and $\boldsymbol{H}^{C,3} = \{0.4, 0.5, 0.6, 0.7\}$, then μ_C is set to 0.4. When $t > t^{\text{learn}}$, elements stored at the earliest iteration are removed from the historical memory (i.e., first-in-first-out replacement policy).

7) PCM-SaNSDE (Algorithm S.17): The PCM of self-adaptive DE with neighborhood search (PCM-SaNSDE) [44] is a hybrid method combining PCM-SaDE (the conference version [43]) and PCM-NSDE [45]. While PCM-SaDE randomly generates the values of F according to the normal distribution, PCM-SaNSDE adaptively selects the probability distribution for generating F as follows:

$$F_{i,t} = \begin{cases} \text{randn}(0.5, 0.3) & \text{if randu}[0, 1] < p \\ \text{randc}(0, 1) & \text{otherwise} \end{cases} \tag{11}$$

where the meta-parameter $p \in [0, 1]$ controls the probability of selecting the normal and Cauchy distributions. The value of p is initialized to 0.5. Then, after the learning period t^{learn}, p is updated as follows:

$$p = \frac{n^{\text{succ1}}\big(n^{\text{total2}}\big)}{n^{\text{succ2}}\big(n^{\text{total1}}\big) + n^{\text{succ1}}\big(n^{\text{total2}}\big)} \tag{12}$$

where n^{total1} and n^{total2} are the number of times that the normal and Cauchy distributions are selected during the learning period. The n^{succ1} and n^{succ2} represent the number of successful trials when using F values generated by each distribution. Once p is updated, the four parameters (n^{total1}, n^{total2}, n^{succ1}, and n^{succ2}) are set to 0.

$C_{i,t}$ is generated according to the procedure of PCM-SaDE, and μ_C is updated based on the weighted mean as follows:

$$\mu_C = \sum_{k=1}^{|\boldsymbol{S}^C|} w_k S_k^C, \;\; w_k = \frac{|f\big(\boldsymbol{x}^k\big) - f\big(\boldsymbol{u}^k\big)|}{\sum_{l=1}^{|\boldsymbol{S}^C|} |f\big(\boldsymbol{x}^l\big) - f\big(\boldsymbol{u}^l\big)|}. \tag{13}$$

8) PCM-JADE (Algorithm S.18): The PCM-JADE [16] uses two adaptive meta-parameters $\mu_F \in (0, 1]$ and $\mu_C \in [0, 1]$ for adaptation of F and C, respectively. At the beginning of the search, μ_F and μ_C are both initialized to 0.5 and adapted during the search. For each iteration t, $F_{i,t}$ and $C_{i,t}$ are generated as follows: $F_{i,t} = \text{randc}(\mu_F, 0.1)$ and $C_{i,t} = \text{randn}(\mu_C, 0.1)$. When $F_{i,t} > 1$, $F_{i,t}$ is truncated to 1. When $F_{i,t} \leq 0$, the new $F_{i,t}$ is repeatedly generated in order to generate a valid value.

For each iteration t, successful F and C parameters are stored into sets $\boldsymbol{S}^F$ and $\boldsymbol{S}^C$, respectively. We use $\boldsymbol{S}$ to refer to $\boldsymbol{S}^F$ or $\boldsymbol{S}^C$ wherever the ambiguity is irrelevant or resolved by context. At the end of the iteration, μ_F and μ_C are updated as: $\mu_F = (1 - c)\ \mu_F + c\ \ \text{mean}_L(\boldsymbol{S}^F)$ and $\mu_C = (1 - c)\ \mu_C + c\ \text{mean}_A(\boldsymbol{S}^C)$, where $c \in [0, 1]$ is a learning rate, $\text{mean}_A(\boldsymbol{S})$ is the arithmetic mean of $\boldsymbol{S}$, and $\text{mean}_L(\boldsymbol{S})$ is the Lehmer mean of $\boldsymbol{S}$ which is computed as: $\text{mean}_L(\boldsymbol{S}) = (\sum_{s \in \boldsymbol{S}} s^2)/(\sum_{s \in \boldsymbol{S}} s)$.

9) PCM-IMDE (Algorithm S.19): The PCM of Islam's modified DE (PCM-IMDE) [33] is similar to PCM-JADE and also uses the meta-parameters μ_F and μ_C. In each iteration t, $F_{i,t}$ and $C_{i,t}$ are generated as same with PCM-JADE. At the end of each iteration, μ_F and μ_C are updated as follows: $\mu_F = (1 - c_F)\ \mu_F + c_F\ \text{mean}_P(\boldsymbol{S}^F)$ and $\mu_C = (1 - c_C)\ \mu_C + c_C\ \ \text{mean}_P(\boldsymbol{S}^C)$, where c_F and c_C are uniformly selected random real numbers from $[0, 0.2]$ and $[0, 0.1]$, respectively. Unlike JADE, the learning rates c_F and c_C are randomly assigned in each iteration t. The function $\text{mean}_P(\boldsymbol{S})$ denotes the power mean of $\boldsymbol{S}$: $\text{mean}_P(\boldsymbol{S}) = ([1/|\boldsymbol{S}|] \sum_{s \in \boldsymbol{S}} s^{1.5})^{(1/1.5)}$.

[1] In the earlier conference version of PCM-SaDE [43], the mean value of elements in $\boldsymbol{H}^C$ is used for update of μ_C.

10) PCM-SHADE (Algorithm S.20): Similar to PCM-SaDE, the PCM-SHADE [15], [46] uses historical memories $\boldsymbol{M}^F$ and $\boldsymbol{M}^C$ for adaption of F and C, where $\boldsymbol{M}^F = (M_1^F, \ldots, M_H^F)$ and $\boldsymbol{M}^C = (M_1^C, \ldots, M_H^C)$. Here, H is a memory size, and all elements in $\boldsymbol{M}^F$ and $\boldsymbol{M}^C$ are initialized to 0.5. Although there have been several slightly different variants of PCM-SHADE [15], [46], [47], the simplest version in [47] is described here. In each iteration t, $F_{i,t}$ and $C_{i,t}$ are generated as follows: $F_{i,t} = \text{randc}(M_{r_{i,t}}^F, 0.1)$ and $C_{i,t} = \text{randn}(M_{r_{i,t}}^C, 0.1)$, where $r_{i,t}$ is a randomly selected index from $\{1, \ldots, H\}$. If $F_{i,t}$ and $C_{i,t}$ are outside the range $[0, 1]$, they are adjusted/regenerated according to the procedure of PCM-JADE.

At the end of each iteration, the memory contents in $\boldsymbol{M}^F$ and $\boldsymbol{M}^C$ are updated using the Lehmer mean as follows: $M_k^F = \text{mean}_L(\boldsymbol{S}^F)$ and $M_k^C = \text{mean}_L(\boldsymbol{S}^C)$. An index $k \in \{1, \ldots, H\}$ determines the position in the memory to update. At the beginning of the search, k is initialized to 1, and incremented whenever a new element is inserted into the history. If $k > H$, k is set to 1.

11) PCM-SLADE (Algorithm S.21): The PCM of SLADE (PCM-SLADE) [48] is similar to PCM-JADE. However, contrary to PCM-JADE, the values of F and C are randomly generated according to the normal and Cauchy distributions as follows: $F_{i,t} = \text{randn}(\mu_F, 0.1)$ and $C_{i,t} = \text{randc}(\mu_C, 0.1)$. When $F_{i,t}$ is outside of $[0, 1]$, it is set to 1. The $C_{i,t}$ is repeatedly generated until it is inside of $[0, 1]$. The arithmetic mean is used for updating both μ_F and μ_C as follows: $\mu_F = (1-c)\ \mu_F + c\ \text{mean}_A(\boldsymbol{S}^F)$ and $\mu_C = (1-c)\ \mu_C + c\ \text{mean}_A(\boldsymbol{S}^C)$. This differs slightly from the original description in [48]. For details, see Section S.2 in the supplementary file.

12) PCM-EPSDE (Algorithm S.22): The PCM of EPSDE (PCM-EPSDE) [22] uses an $\boldsymbol{F}^{\text{pool}}$ and a $\boldsymbol{C}^{\text{pool}}$ for adaptation of F and C, respectively. The $\boldsymbol{F}^{\text{pool}}$ and $\boldsymbol{C}^{\text{pool}}$ are the sets of the F and C values as follows: $\boldsymbol{F}^{\text{pool}} = \{0.4, 0.5, \ldots, 0.9\}$ and $\boldsymbol{C}^{\text{pool}} = \{0.1, 0.2, \ldots, 0.9\}$. At the beginning of the search, each individual $\boldsymbol{x}_{i,t}$ is randomly assigned the values for $F_{i,t}$ and $C_{i,t}$ from each pool. During the search, successful parameter sets are inherited by the individual in the next iteration $t+1$. Parameter sets that fail are reinitialized.

13) PCM-CoBiDE (Algorithm S.23): In the PCM of DE with covariance matrix learning and bimodal distribution parameter setting (PCM-CoBiDE) [49], at the beginning of the search, $F_{i,t}$ and $C_{i,t}$ are randomly sampled according to a bimodal distribution consisting of two Cauchy distributions as follows:

$$F_{i,t} = \begin{cases} \text{randc}(0.65, 0.1) & \text{if randu}[0, 1] < 0.5 \\ \text{randc}(1.0, 0.1) & \text{otherwise} \end{cases} \tag{14}$$

$$C_{i,t} = \begin{cases} \text{randc}(0.1, 0.1) & \text{if randu}[0, 1] < 0.5 \\ \text{randc}(0.95, 0.1) & \text{otherwise} \end{cases} \tag{15}$$

where $F_{i,t}$ and $C_{i,t}$ are outside of $[0, 1]$, they are modified according to the procedure of PCM-JADE. Similar to PCM-EPSDE, a pair of successful F and C values is inherited by each individual in the next iteration. Failed F and C parameter pairs are reinitialized using (14) and (15).

14) PCM-DEDPS (Algorithm S.24): The PCM of DE with dynamic parameters selection (PCM-DEDPS) [50] uses predefined parameter combinations of F and C similar to PCM-cDE, but the pool of parameter combinations is pruned during the search process. For F and C, the following sets of candidate parameters are prepared at the beginning of the search: $\boldsymbol{F}^{\text{pool}} = \{0.4, 0.5, \ldots, 0.9, 0.99\}$ and $\boldsymbol{C}^{\text{pool}} = \{0.2, 0.3, \ldots, 0.9, 0.99\}$. The number of possible combination of the F and C values m is $7 \times 9 = 63$. At the beginning of each iteration, all possible pairs $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^m$ are randomly assigned to all individuals without replacement. If $N > m$, randomly selected pairs are assigned to the $N - m$ individuals.

When t reaches a learning period $t^{\text{CS}} \in \{50, 100, 150, 200\}$, each pair $\boldsymbol{q}^k$ ($k \in \{1, \ldots, m\}$) is ranked based on its score value n_k^{score} calculated as follows: $n_k^{\text{score}} = n_k^{\text{succ}}/n_k^{\text{total}}$. The n_k^{total} is the number of times that $\boldsymbol{q}^k$ was selected, and n_k^{succ} denotes the number of successful trials of $\boldsymbol{q}^k$. A high n_k^{score} indicates that $\boldsymbol{q}^k$ is an appropriate setting. All pairs $\boldsymbol{q}^1, \ldots, \boldsymbol{q}^m$ are sorted based on their score values, and then their lower half is removed from the pool of possible parameter combinations. After the pruning procedure, the two parameters (n_k^{succ} and n_k^{total}) for each $\boldsymbol{q}^k$ are reinitialized to 0.

15) PCM-RDE (Algorithm S.25): In the PCM of rank-based DE (PCM-RDE) [51], for each iteration t, individuals are sorted based on their objective values so that $f(\boldsymbol{x}^{1,t}) \leq \ldots \leq f(\boldsymbol{x}^{N,t})$. Then, $F_{i,t}$ and $C_{i,t}$ are assigned according to the rank value of the base vector $j \in \{1, \ldots, N\}$

$$F_{i,t} = F^{\min} + \left(F^{\max} - F^{\min}\right)\left(\frac{j-1}{N-1}\right) \tag{16}$$

$$C_{i,t} = C^{\max} - \left(C^{\max} - C^{\min}\right)\left(\frac{j-1}{N-1}\right) \tag{17}$$

where $F^{\min}$, $F^{\max}$, $C^{\min}$, and $C^{\max}$ are the minimum and maximum values for F and C, respectively. Their recommended settings are as follows: $F^{\min} = 0.6$, $F^{\max} = 0.95$, $C^{\min} = 0.85$, and $C^{\max} = 0.95$. In (16) and (17), a smaller $F_{i,t}$ value and a larger $C_{i,t}$ value are assigned when the objective value of the base vector is small. While other PCMs described in this section were originally developed for a DE algorithm using binomial crossover, PCM-RDE was designed for a DE with exponential crossover.

16) PCM-IDE (Algorithm S.26): The PCM of DE with an individual dependent mechanism (PCM-IDE) [52] uses the rank values of individuals in the population similar to PCM-RDE. However, while values of F and C are deterministically assigned to individuals based on their ranks in PCM-RDE, they are randomly generated in PCM-IDE. After sorting all individuals based on their objective values, $F_{i,t}$ and $C_{i,t}$ values for an individual $\boldsymbol{x}^{i,t}$ are sampled as follows: $F_{i,t} = \text{randn}(\mu_{F,j}, 0.1)$ and $C_{i,t} = \text{randn}(\mu_{C,i}, 0.1)$, where $\mu_{F,j} = j/N$ ($j \in \{1, \ldots, N\}$) is the rank value of a base vector $\boldsymbol{x}^{j,t}$, and $\mu_{C,i}$ ($i \in \{1, \ldots, N\}$) for the ith ranked individual is i/N. The $F_{i,t}$ and $C_{i,t}$ values are repeatedly generated until they are in $[0, 1]$. In contrast to PCM-RDE, a small $C_{i,t}$ value is assigned to better individuals in the population.

17) PCM-YADE (Algorithm S.27): In the PCM of Yu's adaptive DE (PCM-YADE) [35], for each iteration t, the current search state is classified into the exploration and

exploitation phase based on the distribution of individuals in the objective and solution spaces. Then, $F_{i,t}$ and $C_{i,t}$ are randomly generated according to the current state. The description of the PCM-YADE procedure requires numerous equations—due to space, we show the details in Section S.1 in the supplementary file.

C. SPCMs in DE

1) PCM-SDE (Algorithm S.28): In the PCM of self-adaptive DE (PCM-SDE) [53], $C_{i,t}$ is randomly sampled as follows: $C_{i,t} =$ randn$(0.5, 0.15)$. Unlike C, $F_{i,t}$ is self-adaptively generated as follows: $F_{i,t} = F_{r_1,t} +$ randn$(0, 0.5)(F_{r_2,t} - F_{r_3,t})$. At the beginning of the search, $F_{i,t}$ is initialized according to a normal distribution randn$(0.5, 0.15)$. The indices r_1, r_2, and r_3 are randomly selected from $\{1, \ldots, N\}$ such that they differ from each other. Clearly, the generation method of new F values is identical to the rand/1 mutation strategy. Values of $F_{i,t}$ and $C_{i,t}$ outside of $[0, 1]$ are truncated, e.g., $F_{i,t} = 1.4$ is truncated to 0.4.

IV. Relationships Among PCMs in DE

We discuss the relationships among PCMs for DE, including the 24 PCMs described in Section III. First, we provide a historical overview of the early development of PCMs in Section IV-A. Then, we provide more detailed classifications of PCMs in Section IV-B. DPCMs and SPCMs are described in Sections IV-C and IV-D, respectively.

Then, we turn to the discussions of APCMs, which are currently the most common class of PCMs and the main focus of this section. First, observation-based APCMs are described in Section IV-E. Then, success-based APCMs are discussed in the remaining sections. APCMs with a parameter inheritance mechanism are described in Section IV-F. APCMs with predefined parameters sets are introduced in Section IV-G. PCM-SaDE and PCM-JADE variants are introduced in Sections IV-H and IV-I, respectively. Finally, Section IV-J discusses the PCM-SHADE, which is used in many state-of-the-art DEs and integrates key ideas from the two previously independent lines of work described in Sections IV-H and IV-I.

A. Historical Perspective

Most of PCMs in DE proposed in early studies are derived from those in other EAs, such as GA and PSO. For example, PCM-DERSF, PCM-DETVSF, as well as some other methods (e.g., [54]) are based on PCMs of two PSO variants (random and time-varying inertia weight strategies) [55], [56]. The idea of FADE [57], which adaptively adjusts the F and C parameters using fuzzy logic controllers, is derived from fuzzy GAs [58]. DESAP [59], which adapts the three control parameters (F, C, and the population size N), is also based on self-adaptive GAs.

A significant turning point in the development of PCMs for DE was around 2005. PCM-SaDE [43] (the conference version) and PCM-jDE [12] were proposed in 2005 and 2006, respectively. The two PCMs were designed based on the unique algorithmic characteristics of DE (i.e., the pair-wise survival selection of individuals for the next iteration). PCM-jDE assigns a pair of $F_{i,t}$ and $C_{i,t}$ to each individual $\boldsymbol{x}^{i,t}$, and each parameter is randomly regenerated with a predefined probability as (4) and (5). If the trial vector $\boldsymbol{u}^{i,t}$ is better than its parent individual $\boldsymbol{x}^{i,t}$, the newly generated parameter is inherited by $\boldsymbol{x}^{i,t+1}$. PCM-SaDE samples $C_{i,t}$ for each individual $\boldsymbol{x}^{i,t}$ according to the normal distribution with the mean μ_C and variance 0.1. The parameter μ_C is adaptively updated based on successful C values in the historical memory $\boldsymbol{H}^C$. Recall that the determination of success or failure is made according to a pair-wise comparison between $\boldsymbol{x}^{i,t}$ and $\boldsymbol{u}^{i,t}$. Such APCMs based on the comparison between two individuals had not been proposed for *population-based* EAs.[2]

B. More Detailed Classification of PCMs in DE

Table I shows the properties of the 24 PCMs. Each symbol in Table I is explained below. Although Table I attempts to clearly organize the 24 PCMs reviewed in this paper, we do not claim that this categorization can be applied to all PCMs. A systematic taxonomy which can be used to comprehensively classify the myriad of PCMs for DE in the literature remains an open problem. In Section III, PCMs in DE are classified into DPCMs, APCMs, and SPCMs according to the taxonomy of Eiben *et al.* [1]. Although the classification rule in [1] has been widely accepted in the evolutionary computation community, some previous studies introduce other taxonomies that categorize PCMs for DE in detail.

According to [51], APCMs in DE can be further classified into the following two categories: 1) "S" success-based control and 2) "O" observation-based control. While success-based APCMs adjust parameter values of F and C based on the success/failure decision, and observation-based APCMs adapt parameter values using some indicator values of individuals in the solution and/or objective spaces.

Another taxonomy in [23] categorizes PCMs based on the following four factors: 1) the type of parameter values ("C" continuous or "D" discrete values); 2) the number of parameter values for each iteration ("S" a single value or "M" multiple values); 3) the information used while sampling new parameter values; and 4) the parameter inheritance mechanism. For the third factor, the following four information resources can be considered: 1) "N" no information; 2) "T" time; 3) "P" the distribution of individuals in the population; and 4) "H" the historical information obtained during the search process. Note that this description of their classification differs slightly from the original one in [23]. Since some factors in [23] can be represented by the taxonomy in [1], we modified the original taxonomy for the sake of clarity.

As seen from Table I, while most APCMs can be categorized as either a success-based or an observation-based control method, PCM-FDSADE and PCM-ISADE are exceptional and belong to both categories. Although PCM-FDSADE and PCM-ISADE sample new F and C values based on the objective values of individuals, they use the success/failure

[2]The step-size adaptation of one-fifth success rule for $(1 + 1)$-ES [4] is also based on the pair-wise comparison between a parent individual $\boldsymbol{x}^t$ and a child $\boldsymbol{u}^t$. However, $(1 + 1)$-ES uses only one individual for the search.

determination to select F and C values which are inherited to the next iteration. Five PCMs (PCM-CoDE, PCM-SWDE, PCM-cDE, PCM-EPSDE, and PCM-DEDPS) use predefined discrete parameter values. While a single parameter value is commonly used in all the individuals in three PCMs (PCM-DETVSF, PCM-SinDE, and PCM-DEPD), the remaining 21 PCMs use different values of F and C when producing offspring in an iteration. Two DPCMs (PCM-DETVSF and PCM-SinDE) are time-dependent PCMs, and other PCMs utilize various types of information to sample new parameter values. In five APCMs (three PCM-jDE and two PCM-EPSDE variants) and one SPCM (PCM-SDE), the F and C values used in each individual can be inherited by the next iteration.

C. DPCMs

PCM-DERSF and PCM-DETVSF, which are classified as DPCMs, were proposed in 2005. However, DPCMs did not become popular in the DE community until PCM-CoDE was proposed in 2011. CoDE is a simple but efficient DE algorithm with a DPCM, and experimental results in [17] show that it performs better than four DE algorithms with APCMs. After the proposal of PCM-CoDE, some efficient DPCMs (e.g., PCM-ZMDE, PCM-SinDE, and PCM-SWDE) were proposed. While PCM-DERSF and PCM-DETVSF adjust only values of F, four DPCMs (PCM-CoDE, PCM-ZMDE, PCM-SinDE, and PCM-SWDE) control both F and C parameters. Moreover, PCM-CoDE, PCM-SinDE, and PCM-SWDE generate values of F and C in a wider range than PCM-DERSF and PCM-DETVSF. PCM-CoDE and PCM-SWDE are similar in that they randomly assign a predefined parameter pair $\{F, C\}$ to each individual (see Table I). It is interesting to note that an adaptive version of PCM-CoDE based on the PCM-SaDE scheme is proposed in [17]. However, it performs significantly worse than the simple, original PCM-CoDE in the same DE framework.

D. SPCMs

SPCMs have not received much attention in the DE community, so only one SPCM (PCM-SDE) is described in Section III-C. In addition to PCM-SDE, PCMs in "SPDE" [60], "DESAP" [59], "ESADE" [61], and "PBMODE" [62] can be classified into SPCMs.[3] However, these algorithms do not fit well within the general DE framework covered in this paper. Unlike a standard DE, GA mutation operators are incorporated into SPDE and DESAP. The PCM of ESADE needs to generate two trial vectors for each individual. A Pareto-dominance relation is used in the PCM of PBMODE. Thus, it is difficult to extract only the PCM from SPDE, DESAP, ESADE, and PBMODE.

E. Observation-Based APCMs

Observation-based APCMs use indicator values from the individuals and/or their objective values for adaptation of F and C. In addition to the four APCMs in Table I, other observation-based APCMs have been proposed, including PCM-FADE [57] and PCM-FiADE [64].

Interestingly, each observation-based APCM was designed based on different search policies. While PCM-DEPD, PCM-FiADE, PCM-RDE, and PCM-IDE use only information from the objective space, PCM-FADE and PCM-YADE assign values of F and C to individuals based on their diversity in both objective and solution spaces. In PCM-RDE and PCM-FiADE, small C values are assigned to each individual in case of a selected base vector is *inferior* in the population. In contrast, PCM-IDE assigns small C values to *superior* individuals. Thus, the key to design an efficient observation-based APCM is how the information obtained from the distribution of individuals in the population is used.

F. APCMs With Parameter Inheritance Mechanism

Some variants of PCM-jDE have been proposed, such as PCM-FDSADE and PCM-ISADE, as well as other methods (e.g., [65]). Although $F_{i,t}$ and $C_{i,t}$ for $\boldsymbol{x}_{i,t}$ are probabilistically changed to new values in PCM-jDE variants, this probability and the method of generating values of F and C are different for each PCM, as shown in (4)–(9). While the probability is constant in PCM-jDE, it depends on the diversity of the objective values of individuals in PCM-FDSADE. PCM-ISADE utilizes the objective value of each individual to determine the probability and generate new values of F and C.

Since the parameter assignment strategy of PCM-CoBiDE is identical with that of PCM-EPSDE, PCM-CoBiDE is considered to be a variant of PCM-EPSDE. While PCM-EPSDE uses a predefined parameter pool, PCM-CoBiDE generates the parameters with a bimodal distribution as (14) and (15). In addition to PCM-CoBiDE, some PCM-EPSDE variants have been proposed (e.g., [66]).

Although PCM-jDE and PCM-EPSDE are similar in that they assign parameter values of F and C to each individual, their adaptation strategies differ. While PCM-jDE uses new parameters probabilistically, PCM-EPSDE assigns new parameter values when a trial vector generation fails. An analysis in [47] shows that there is a significant difference in the parameter adaptation ability of PCM-jDE and PCM-EPSDE.

G. APCMs Using Predefined Discrete Parameters Sets

In three APCMs (PCM-cDE, PCM-EPSDE, and PCM-DEDPS), discrete values of F and C are defined before the search and adaptively assigned to each individual in the population. However, their parameter adaptation mechanisms are significantly different from each other. PCM-cDE assigns each pair of parameter values to individuals with a probability based on the number of its successful trials as (10). PCM-EPSDE randomly assigns a pair of F and C values to an individual at the beginning of the search. The parameter values for each individual are reassigned when the trial is a failure. On each iteration, PCM-DEDPS assigns all the combinations of F and C values uniformly to all the individuals in the population. Then, parameter combinations which perform poorly regarding the number of successful trials are removed periodically.

[3] A method of adjusting a weight factor used in a mutation strategy in DEGL [63] belongs to SPCMs.

Since this kind of APCMs relies entirely on the predefined set of discrete parameters for adaptation, their performance is influenced by the elements of these predefined parameters sets. The effect of the parameters set in PCM-DEDPS is investigated in [50]. PCM-cDE and PCM-EPSDE variants with other parameter sets have been proposed (e.g., [28] and [67]).

H. PCM-SaDE Variants

As mentioned in Section IV-A, PCM-SaDE is one of the earliest APCMs in DE. Nevertheless, only a few variants of PCM-SaDE (i.e., PCM-SaNSDE and PCM-SHADE) have been proposed. This may be because the mutation strategy adaptation in "SaDE" attracted more attention than its parameter adaptation mechanism.

PCM-SaNSDE [44] is an extended version of the conference version of PCM-SaDE [43]. While PCM-SaDE randomly generates the values of F according to the normal distribution, PCM-SaNSDE adaptively selects the probability distribution for generating F. Interestingly, it is pointed out in [68] that the adaptive scheme of PCM-SaNSDE for F in (11) can be replaced with a simple deterministic generation method on some problems without performance loss.

Similar to PCM-SaDE, PCM-SHADE uses the historical memory for adaptation of F and C. However, the memory update rule and the method for generating F and C values in PCM-SHADE are different from that of PCM-SaDE.

I. PCM-JADE Variants

PCM-JADE has had a significant impact on the development of PCMs in DE. In addition to the three APCMs (PCM-IMDE, PCM-SHADE, and PCM-SLADE) reviewed in Section III-B, a number of PCM-JADE variants have been proposed (e.g., PCM-GaDE [69], PCM-MADE [70], PCM-DMPSADE [71], and PCM-ZEPDE [72]).

PCM-JADE variants use meta-parameters, such as μ_F and μ_C as the mean/location parameter values of random distributions to generate F and C values. The meta-parameters are updated based on successful F and C values at the end of each iteration. While PCM-JADE uses a constant learning rate c, PCM-IMDE randomly generates c_F and c_C values for F and C in predefined intervals, respectively. Unlike PCM-JADE, PCM-SLADE uses the normal and Cauchy distributions for generating the values of F and C, respectively.

J. PCM-SHADE in State-of-the-Art DEs

PCM-SHADE combines two key ideas from the lines of work described above: the use of a success-history memory from PCM-SaDE (Section IV-H), and the idea from PCM-JADE of generating new values for F and C by sampling regions centered around previously good values (Section IV-I). The successful combination of these previously disparate lines of work led to a significant improvement in the performance of state-of-the-art DE algorithms. Since 2014, DEs with PCM-SHADE (e.g., L-SHADE [15]) have dominated the annual IEEE CEC competitions on single-objective continuous optimization (http://www3.ntu.edu.sg/home/EPNSugan/index_files/). A number of high-performance DE algorithms with PCM-SHADE have been proposed in the past several years [73].

V. Benchmarking PCMs

In this section, we investigate the performance of the 24 PCMs described in Section III. In general, it is difficult to directly evaluate the parameter control ability of PCMs. Therefore, we evaluate the search performance of DE algorithms using various operators and the 24 PCMs, and treat these results as a proxy for the performance of the 24 PCMs. To implement these simplified DE algorithms, we started with the basic, baseline DE [6] described in Section II and made the minimal modifications to accommodate each PCM. For the detail of each DE, see Algorithms S.5–S.28 in the supplementary file. Below, we refer to "the search performance of a complex DE with a PCM" as "the performance of a PCM" for simplicity. Also, "a complex DE with a PCM" and "a PCM" are used synonymously in this section.

Section V-A describes the experimental settings. Section V-B compares the performance of PCMs using hyperparameter settings recommended in the literature (usually the original paper), while Section V-C compares the performance of PCMs using hyperparameter settings obtained using SMAC [25]. Section V-D investigates the gap between the performance of existing PCMs with that of a lower bound on the performance of an "optimal" PCM obtained by greedy approximate oracle (GAO) method [26].

A. Experimental Settings

All experiments were conducted using the COCO software (http://coco.gforge.inria.fr/), the standard benchmarking platform used in the BBOB workshops held at GECCO (2009-present) and CEC (2015). We used the noiseless BBOB benchmark set [24] consisting of 24 test functions $f_1, \ldots, f_{24}$. The 24 BBOB functions are grouped into five categories: 1) separable functions ($f_1, \ldots, f_5$); 2) functions with low or moderate conditioning ($f_6, \ldots, f_9$); 3) functions with high conditioning and unimodal ($f_{10}, \ldots, f_{14}$); 4) multimodal functions with adequate global structure ($f_{15}, \ldots, f_{19}$); and 5) multimodal functions with weak global structure ($f_{20}, \ldots, f_{24}$). The dimensionality D of the functions was set to 2, 3, 5, 10, 20, and 40. For each problem instance, 15 runs were performed. These settings adhere to the standard benchmarking and analysis procedure adopted by the BBOB community in the BBOB workshops since 2009. The maximum number of FEvals was $10\,000 \times D$.

In addition to the 24 PCMs, we evaluated the performance of the DE without any PCMs [6]. The F and C parameters were set to 0.5 and 0.9, respectively. These parameter settings are commonly used in the previous work (e.g., [12] and [16]). Below, the DE with $F = 0.5$ and $C = 0.9$ is denoted as "$F05C09$."

Following the previous work [74], the population size N was set to $5 \times D$ for $D \geq 5$, and 20 for $D \leq 3$. We evaluated the performance of the 25 methods (the 24 PCMs and $F05C09$) using eight different mutation operators: 1) rand/1; 2) rand/2; 3) best/1; 4) best/2; 5) current-to-rand/1; 6) current-to-best/1; 7) current-to-$p$best/1; and 8) rand-to-$p$best/1. Details of the

mutation operators are in Table S.1 in the supplementary file. The control parameters of the current-to-$p$best/1 and rand-to-$p$best/1 strategies were set to $p = 0.05$ and $|\boldsymbol{A}| = N$ as in [16]. We evaluated both bin and sec. In summary, our benchmarking study investigated the performance of the 25 methods using 16 variation operators (8 mutation strategies $\times$ 2 crossover methods).

Restarts are standard practice in the BBOB community and the COCO software documentation recommends the use of restarts, so we incorporated the restart strategy of [75] (a slightly modified version of the method in [76]) into all methods, except for PCM-SinDE and PCM-DETVSF.[4] Section S.3 in the supplementary file describes the restart strategy used.

B. Performance of PCMs With Default Hyperparameters

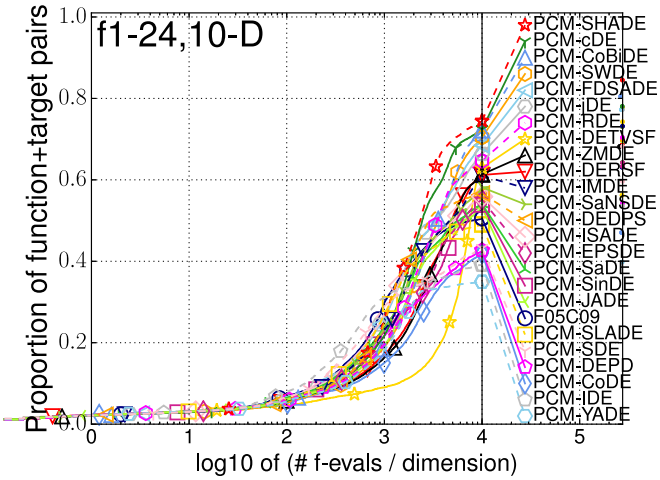
Fig. 1 compares the 25 methods using the rand/1/bin and current-to-$p$best/1/bin operators on the 10-D BBOB benchmark set. Fig. 1 shows the bootstrapped empirical cumulative distribution function (ECDF) [77] of the number of FEvals divided by dimension for 51 targets in $10^{[-8..2]}$ for all 24 BBOB functions. We used the COCO software to generate the ECDF figures. The results of the 25 methods using all 16 operators on the 24 BBOB functions with $D \in \{2, 3, 5, 10, 20, 40\}$ are in Figs. S.1–S.16 in the supplementary file. In this experiment, hyperparameter settings which are recommended in each paper were used for each PCM. For details, see Tables S.2–S.26 in the supplementary file.

In Fig. 1, the vertical axis "proportion of function + target pairs" indicates the proportion of target objective values, in which a given algorithm can reach within specified evaluations. For example, in Fig. 1(b), PCM-CoDE reaches about 20% of all target values within $1000 \times D$ evaluations. If an algorithm finds the optimal solutions on all 24 functions in all 15 runs, the vertical value becomes 1. A more detailed explanation of the ECDF is in Section S.4 in the supplementary file.

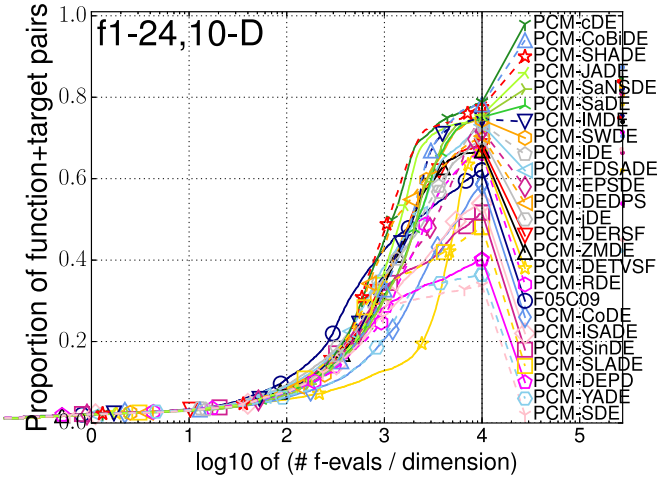
1) Statistical Analysis: In addition to ECDF figures, Figs. S.17–S.32 in the supplementary file show the average performance score (APS) [78] based on the error value $|f(\boldsymbol{x}^{\text{bsf}} - f(\boldsymbol{x}^*)|$ of the 25 methods for all 24 functions, where $\boldsymbol{x}^{\text{bsf}}$ is the best-so-far solution found during the search process, and $\boldsymbol{x}^*$ is the optimal solution of a given problem. The APS value was calculated using the Wilcoxon rank-sum test with $p < 0.05$. For details of the APS, see Section S.5 in the supplementary file. Since there is no significant difference between results based on the ECDF and the APS, we discuss the performance of the 25 methods based only on the ECDF figures.

Recall that our interest is in the performance of PCMs, not the performance of their original, complex DE algorithms. Also, we are not interested in which algorithmic configuration (i.e., a particular combination of a PCM and a variation operator) performs the best. When we say below, e.g., "PCM1 performs better than PCM2," such claims are limited to the PCMs as implemented in our standardized, experimental harness. Thus, we make no claims as to whether the original, complex DE algorithm with PCM1 performs better than that with PCM2.

[4]In our preliminary experiments, PCM-SinDE and PCM-DETVSF without the restart strategy performed better than with the restart strategy. PCM-SinDE and PCM-DETVSF gradually adjust control parameters according to the number of iterations, but the restart strategy disturbs their control schedules by reinitializing the population. For this reason, time-dependent PCMs are incompatible with the restart strategy.



(a)



(b)

Fig. 1. Comparisons of the 25 methods (the 24 PCMs and $F05C09$) with the rand/1/bin and current-to-$p$best/1/bin operators on the 10-D BBOB benchmark set (higher is better).

Fig. 1(a) shows that PCM-IDE has good performance on the 10-D BBOB functions within $1000 \times D$ evaluations when using the rand/1/bin operator. Beyond $2000 \times D$ evaluations, PCM-SHADE performs best. Fig. 1(b) indicates that $F05C09$ (the static parameter setting) outperforms all 24 PCMs within $800 \times D$ evaluations when using the current-to-$p$best/1/bin operator. For more than $1000 \times D$ evaluations, PCM-cDE, PCM-CoBiDE, and PCM-SHADE perform well. The performance of PCM-DETVSF dramatically improved at the end of the search due to its time-dependent control strategy. Due to space constraints, detailed results for the other DE operators are described in Section S.6 in the supplementary file.

TABLE II
BEST-PERFORMING PCM FOR EACH DE OPERATOR ON THE BBOB PROBLEMS (SUMMARY DATA; SEE SUPPLEMENTARY FILE FOR DETAILS). FOR EACH DIMENSIONALITY D AND FOR EACH PCM, WE LIST THE CASES WHERE THE COMBINATION OF THAT PCM AND A DE OPERATOR O PERFORMED BEST (AMONG ALL COMBINATIONS OF O WITH THE ALL TESTED PCMs). THE NUMBERS STAND FOR: (1) RAND/1, (2), RAND/2, (3) BEST/1, (4) BEST/2, (5) CURRENT-TO-RAND/1, (6) CURRENT-TO-BEST/1, (7) CURRENT-TO-pBEST/1, AND (8) RAND-TO-pBEST/1. ALSO, (B) AND (S) DENOTE THE BINOMIAL AND SHUFFLED EXPONENTIAL CROSSOVER METHODS, RESPECTIVELY. THEIR COMBINATIONS REPRESENT THE 16 DE OPERATORS, E.G., (3 S) INDICATES THE BEST/1/SEC OPERATOR. FOR EXAMPLE, FOR $D = 10$, FOR BOTH CURRENT-TO-pBEST/1/BEST/SEC AND RAND-TO-pBEST/1/SEC OPERATORS, PCM-CDE PERFORMS BEST

	$D = 2$	$D = 3$	$D = 5$	$D = 10$	$D = 20$	$D = 40$
$F05C09$	1b, 2b, 3b, 6b, 7b, 8b, 1s, 3s, 4s, 8s	4b, 8b, 2s, 4s, 7s, 8s	4s			
PCM-SinDE	2s, 7s				2s	
PCM-cDE	4b, 5b, 5s, 6s		8b	7s, 8s	1s, 5s, 7s, 8s, 6s	5b, 7b, 5s, 7s, 8s
PCM-ISADE						2b
PCM-JADE					6b	6b
PCM-SHADE		1b, 2b, 5b, 1s, 3s, 5s	1b, 2b, 4b, 5b, 1s, 2s, 5s	1b, 2b, 4b, 5b, 6b, 8b, 1s, 2s, 4s, 5s, 6s	1b, 2b, 3b, 4b, 5b, 7b, 8b	1b, 3b, 4b, 8b, 6s
PCM-CoBiDE		3b, 6b, 7b, 6s	3b, 7b, 3s, 8s	3b, 7b, 3s	3s, 4s	3s, 4s
PCM-IDE			6b, 6s, 7s			1s, 2s

Table II summarizes the best-performing PCM for each operator on the 24 BBOB functions with each dimensionality D, with respect to the APS value at the end of the search. Interestingly, Table II shows that the baseline, static parameter setting ($F = 0.5$ and $C = 0.9$) is highly competitive with the 24 PCMs on low-dimensional problems ($D \in \{2, 3\}$) for most operators. Thus, the fixed parameter setting is suitable for low-dimensional problems. PCM-SHADE performs well for most variation operators for $D \in \{3, 5, 10, 20\}$, and PCM-CoBiDE is competitive with PCM-SHADE. For $D \geq 20$, PCM-cDE is the most suitable for five DE operators. Also, PCM-ISADE, PCM-JADE, and PCM-IDE have the best performance for a particular operator for $D = 40$. These observations suggest that PCMs other than PCM-SHADE are likely to work well for high-dimensional problems. In addition, Table II indicates that there are certain combinations of PCM and DE operator which tend to perform particularly well together. For example, for the best/1/sec operator, PCM-CoBiDE has the best performance for all $D \geq 5$.

Overall, we observed that the performance rankings of the PCMs depend significantly on the dimensionality of the target problems, resource budget, and the types of variation operators used. The dependence of PCM performance on various factors has not been investigated in depth in the literature.

Observation 1: The performance of the PCMs relative to each other depends significantly on: 1) the dimensionality of the target problems; 2) available budget (FEvals); and 3) the types of DE operators.

Our results provide useful insights which may guide the improvement of DEs. Cooperative co-evolution DEs that decompose a given problem into multiple lower-dimensional subproblems are popular approaches for high-dimensional continuous optimization [79]. The same DE (e.g., "SaNSDE" in [79]) is usually applied to each decomposed subproblem in the general framework. However, it may be better to use DEs with different PCMs for each subproblem according to their dimensionality D (e.g., $F05C09$ for 2-D and 3-D subproblems and PCM-cDE for 40-D subproblems). Our results show that the best PCM depends on the three factors (see Observation 1). Thus, a design of a hyper-heuristic [80] method that adaptively selects an appropriate PCM during the search process is an interesting direction for future work.

Although it is difficult to determine a single "best" PCM that works well for all 16 DE operators due to the abovementioned reason, PCM-cDE, PCM-SHADE, and PCM-CoBiDE perform well for most of the 16 variation operators on almost all problems. While only PCM-RDE was originally designed for DE algorithms using (shuffled) exponential crossover, PCM-cDE, PCM-SHADE, and PCM-CoBiDE perform well with both *bin* and *sec*. In an approximated optimal parameter adaptation process which was found in [26], values of F and C are generated in the extreme regions $[0, 0.1]$ and $[0.9, 1]$. PCM-cDE and PCM-CoBiDE are also capable of generating parameter values in such extreme regions. PCM-SHADE stores a diverse set of control parameters in the historical memories $\boldsymbol{H}^F$ and $\boldsymbol{H}^C$, which potentially allow to sample the parameter values in the extreme regions. This is likely the reason why the three PCMs show good performance among the 25 methods.

Observation 2: PCM-cDE, PCM-SHADE, and PCM-CoBiDE perform well for most of the 16 variation operators on almost all problems when using the hyperparameter settings recommended in the literature.

C. Performance of PCMs With Tuned Hyperparameters

As described in Section III, most of PCMs have some hyperparameters for controlling the F and C parameters (e.g., τ_F and τ_C for PCM-jDE). The hyperparameter settings recommended in the literature were used for each PCM in the performance comparison presented in Section V-B. However, in general, the performance of PCMs depends significantly on their hyperparameter settings [16], [81]. Some PCMs (i.e., PCM-DERSF, PCM-DETVSF, and PCM-DEPD) also require a static, fixed C value. Although the population size N was set to $5 \times D$ for all methods in Section V-B, a suitable N value differs from each method [82]. Thus, the parameter settings recommended in the literature for each method may not be the most appropriate values for our benchmark problems.

In this section, we investigate the performance of PCMs with tuned the parameter settings found by an automatic algorithm configurator. In our benchmarking study, we used SMAC [25], which is a surrogate-model-based configurator.

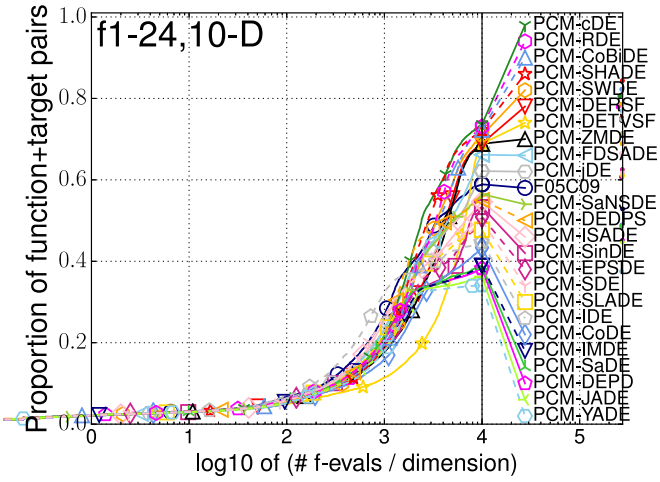
SMAC can be used to tune real-valued, integer-valued, categorical, and conditional parameters. We used the latest version of SMAC (version 2.10.03) downloaded from the authors' website (http://www.cs.ubc.ca/labs/beta/Projects/SMAC/).

The evaluation function used by SMAC to assess the quality of a candidate DE configuration was the mean of the difference between the objective value of the best-so-far solution found by the DE configuration and the optimal value for each training problem. We used 16 functions $f_1^{\text{cec14}}, \ldots, f_{16}^{\text{cec14}}$ with $D \in \{2, 10, 20\}$ from the CEC2014 benchmarks [83] as the training problem set. We excluded $F_{17}^{\text{cec14}}, \ldots, F_{30}^{\text{cec14}}$ because their 2-D versions were not included in the CEC2014 benchmarks. Each run of SMAC was limited to 3000 DE configurations. For each PCM as well as the static DE without any PCM, three independent SMAC runs were performed. Then, for each method, we selected the best parameter settings from the three configurations obtained by SMAC and the recommended parameter setting.

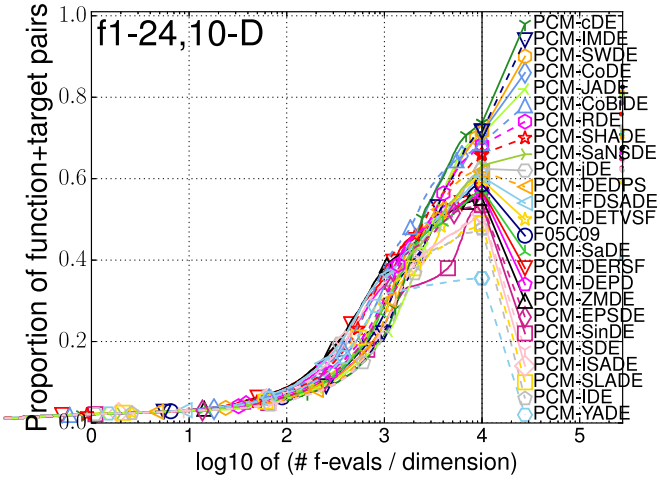
Tables S.2–S.26 in the supplementary file show the search range of each parameter, three configurations found by SMAC, and the best parameter setting for each method. Since the automated parameter configuration of the 25 methods required high computational cost, we conducted their parameter tuning only for the following four representative operators: 1) rand/1/bin; 2) rand/1/sec; 3) current-to-*p*best/1/bin; and 4) current-to-*p*best/1/sec. The rand/1 operator is the most basic mutation strategy, and current-to-*p*best/1 [16] is one of the most commonly used scheme in the recent works (e.g., [33], [35], [46], [49], [50], and [52]).

For some PCMs, parameter settings found by SMAC differ significantly from the recommended values. For example, for PCM-JADE (Table S.16 in the supplementary file), while its default values of μ_F and μ_C are 0.5, the tuned values found by SMAC for the rand/1/sec operator take extreme values (0.15 and 1, respectively). However, for some PCMs (e.g., PCM-cDE as shown in Table S.13 in the supplementary file), SMAC could not find parameter settings which were better than the default settings. The reason for this may be that many of the recommended parameter settings for the PCMs have already been tuned by the original authors on standard benchmark problems similar to those in the BBOB benchmarks, as well as benefitting from the community knowledge acquired in large-scale parameter studies, such as [10]–[12], making it difficult to obtain further improvement using current algorithm configurators.

Fig. 2 compares DE algorithms using the rand/1/sec operator with the 25 methods on the 10-D BBOB benchmark set. Other results can be found in Figs. S.33–S.36 in the supplementary file. Figs. S.37–S.40 in the supplementary file also show the APS of the 25 methods for all 24 BBOB functions. Fig. 2(a) and (b) shows the experimental results for the 25 methods with the default and tuned parameter configurations, respectively. The performance of some PCMs is improved after the parameter tuning. For example, while PCM-JADE, PCM-IMDE, and PCM-CoDE with the default parameter settings does not work well [Fig. 2(a)], they with the tuned configurations have good performance at the end of the search [Fig. 2(b)]. In summary, the results indicate



(a)



(b)

Fig. 2. Comparisons of the 25 methods (the 24 PCMs and *F*05*C*09) using the rand/1/sec operator on the BBOB benchmarks ($D = 10$). (a) and (b) Experimental results of the 25 methods with the default and tuned parameter configurations.

that the performance of some PCMs can be improved by hyperparameter tuning.

However, the results significantly depend on the variation operator used and the dimensionality of the target problems. Also, for most of the cases, the three PCMs (PCM-cDE, PCM-SHADE, and PCM-CoBiDE), which had good performance in the experiments using recommended parameter settings (see Section V-B), still perform well among the 25 methods in this experiment (see Figs. S.33–S.36 in the supplementary file).

Observation 3: PCM-cDE, PCM-SHADE, and PCM-CoBiDE perform well for most of the four variation operators on almost all problems using parameter settings obtained using an automatic algorithm configurator.

D. Is There Still Room for Significant Improvement of PCMs?

Next, we compare the performance of existing PCMs with that of the GAO method [26], an oracle-based method

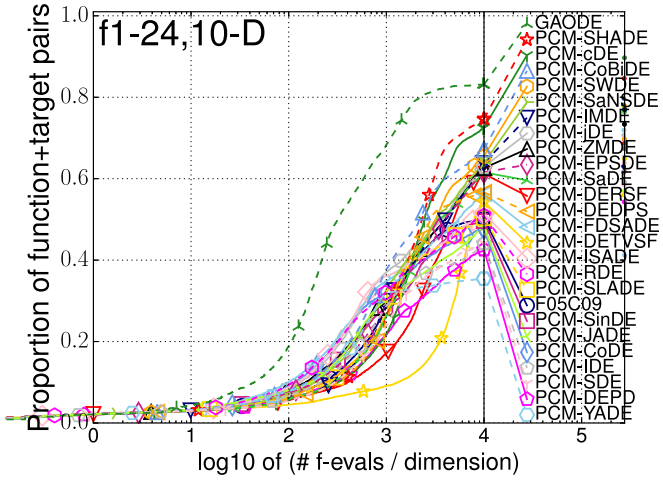


Fig. 3. Comparison of the 25 methods with GAODE on the BBOB benchmarks ($D = 10$). Tuned hyperparameter settings were used for each PCM. The rand/1/bin operator was used for all methods.

for obtaining a (lower) bound on optimal PCM behavior. Such comparison give some indication of how much room there is for further improvement beyond the current state-of-the-art.

GAO is a simulation-based method for *approximating* an optimal parameter adaptation process. For each step of the search (i.e., the parameter sampling of $F_{i,t}$ and $C_{i,t}$ for the individual $\boldsymbol{x}^{i,t}$), GAO randomly samples many possible control parameter sets to *retrospectively* identify a control parameter set which would have yielded the best-expected result (with respect to 1-step-lookahead) on that step. By repeating this process until the search terminates, GAO obtains a parameter adaptation process that is approximately optimal with respect to 1-step-lookahead. It should be emphasized that GAO is a *tool* for understanding the limitation of APCMs, not an actual APCM. Below, "GAODE" denotes a DE that incorporates GAO to adjust values of $F_{i,t}$ and $C_{i,t}$.

Fig. 3 shows the comparison of the 25 methods with GAODE on the 10-D BBOB functions. Detailed results on the 24 BBOB functions with $D \in \{2, 3, 5, 20\}$ are in Fig. S.41 in the supplementary file. The experimental data for GAODE was derived from [26]. Tuned hyperparameter settings were used for each PCM. Since GAO *currently* works well only with the rand/1/bin operator, the comparison was conducted only for DE algorithms with the rand/1/bin.

As shown in Fig. 3, all of the 25 methods are significantly outperformed by GAODE. For example, although PCM-SHADE performs best among the 25 methods when using rand/1/bin on the 10-D problems, its performance is about ten times worse than that of GAODE in terms of convergence speed (Fig. 3). The results on other dimensional problems in Fig. S.41 in the supplementary file are similar. Furthermore, note that GAODE is only a 1-step greedy approximation of an optimal PCM process—a true optimal PCM process should significantly outperform GAODE. In summary, while many PCMs have been proposed in the DE community, the comparison with GAODE indicates that there is much room for development of more efficient PCMs.

Observation 4: Even the best current PCMs converge at least ten times slower than a DE which has an access to an oracle that provides perfect PCM behavior.

VI. Conclusion

The contributions of this paper are threefold. The first is an in-depth review of 24 PCMs for DE. We extracted the PCM components from DE algorithms and precisely described in a unified framework using common terminology. Our review provides a systematic classification and characterization of PCMs for DE, facilitating the understanding of similarities/differences between the numerous proposed PCMs.

The second contribution is a large-scale, benchmarking study of the 24 PCMs (and a DE with fixed parameter settings) using 16 DE operators on the BBOB benchmarks [24]. We investigated the performance of the PCMs with both recommended as well as tuned hyperparameter settings. We observed that although the relative performances of the PCMs depend significantly on three factors (the dimensionality of problems, available budget, and the variation operator used), PCM-cDE, PCM-SHADE, and PCM-CoBiDE exhibited consistently high performance.

The third contribution is an assessment of how far the state-of-the-art PCMs are from an ideal PCM. We compared the 24 PCMs with the oracle-based GAODE model [26]. The results show that the 24 PCMs perform significantly worse than GAODE and, thus, there is still much room for the future work in the development of novel, efficient PCMs.

We focused on PCMs for the F and C parameters. In addition to F and C, various PCMs have applied parameter control for other aspects of the DE, such as mutation strategies (e.g., [17], [20], and [22]) and the population size N (e.g., [15], [21], [36], [59], and [76]). Reviewing and benchmarking these other types of PCMs is an avenue for the future work.

References

[1] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, Jul. 1999.
[2] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 167–187, Apr. 2015.
[3] T. C. Fogarty, "Varying the probability of mutation in the genetic algorithm," in *Proc. ICGA*, 1989, pp. 104–109.
[4] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution*. Stuttgart, Germany: Frommann-Holzboog, 1973.
[5] T. Bäck and H. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Comput.*, vol. 1, no. 1, pp. 1–23, Mar. 1993.
[6] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[7] K. V. Price, R. N. Storn, and J. A. Lampinen, "Differential evolution: A practical approach to global optimization," in *Natural Computing Series*. Heidelberg, Germany: Springer, 2005.
[8] 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.
[9] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution—An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, Apr. 2016.

[10] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *Proc. Int. Conf. Adv. Intell. Syst. Fuzzy Syst.*, 2002, pp. 293–298.

[11] K. Zielinski, P. Weitkemper, R. Laur, and K.-D. Kammeyer, "Parameter study for differential evolution using a power allocation problem including interference cancellation," in *Proc. IEEE CEC*, 2006, pp. 1857–1864.

[12] 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 Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[13] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Art. Intell. Rev.*, vol. 33, nos. 1–2, pp. 61–106, 2010.

[14] E.-N. Dragoi and V. Dafinescu, "Parameter control and hybridization techniques in differential evolution: A survey," *Artif. Intell. Rev.*, vol. 45, no. 4, pp. 447–470, 2016.

[15] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE CEC*, 2014, pp. 1658–1665.

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

[17] 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.

[18] K. Zielinski, X. Wang, and R. Laur, "Comparison of adaptive approaches for differential evolution," in *Proc. PPSN*, 2008, pp. 641–650.

[19] M. Drozdik, H. E. Aguirre, Y. Akimoto, and K. Tanaka, "Comparison of parameter control mechanisms in multi-objective differential evolution," in *Proc. LION*, 2015, pp. 89–103.

[20] 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.

[21] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Appl. Intell.*, vol. 29, no. 3, pp. 228–247, 2008.

[22] 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, 2011.

[23] T.-C. Chiang, C.-N. Chen, and Y.-C. Lin, "Parameter control mechanisms in differential evolution: A tutorial review and taxonomy," in *Proc. IEEE SDE*, 2013, pp. 1–8.

[24] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Rocquencourt, France, Rep. RR-6829, 2009.

[25] F. Hutter, F. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proc. LION*, 2011, pp. 507–523.

[26] R. Tanabe and A. Fukunaga, "How far are we from an optimal, adaptive DE?" in *Proc. PPSN*, 2016, pp. 145–155.

[27] J. Brest, B. Boškovič, S. Greiner, V. Žumer, and M. S. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Comput.*, vol. 11, no. 7, pp. 617–629, 2007.

[28] J. Tvrdík, "Adaptation in differential evolution: A numerical comparison," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1149–1155, 2009.

[29] C. Segura, C. A. C. Coello, E. Segredo, and C. León, "On the adaptation of the mutation scale factor in differential evolution," *Opt. Lett.*, vol. 9, no. 1, pp. 189–198, 2015.

[30] C. Segura, C. A. C. Coello, E. Segredo, and C. León, "An analysis of the automatic adaptation of the crossover rate in differential evolution," in *Proc. IEEE CEC*, 2014, pp. 459–466.

[31] J. Zhang and A. Sanderson, *Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization*, vol. 1. Heidelberg, Germany: Springer, 2009.

[32] R. Tanabe and A. Fukunaga, "Reevaluating exponential crossover in differential evolution," in *Proc. PPSN*, 2014, pp. 201–210.

[33] 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.

[34] D. Zou, J. Wu, L. Gao, and S. Li, "A modified differential evolution algorithm for unconstrained optimization problems," *Neurocomputing*, vol. 120, pp. 469–481, Nov. 2013.

[35] W. Yu *et al.*, "Differential evolution with two-level parameter adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1080–1099, Jul. 2014.

[36] V. Tirronen and F. Neri, "Differential evolution with fitness diversity self-adaptation," in *Nature-Inspired Algorithms for Optimisation*. Heidelberg, Germany: Springer, 2009, pp. 199–234.

[37] L. Jia, W. Gong, and H. Wu, "An improved self-adaptive control parameter of differential evolution for global optimization," in *Proc. ISICA*, 2009, pp. 215–224.

[38] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. GECCO*, 2005, pp. 991–998.

[39] A. Draa, S. Bouzoubia, and I. Boukhalfa, "A sinusoidal differential evolution algorithm for numerical optimisation," *Appl. Soft Comput.*, vol. 27, pp. 99–126, Feb. 2015.

[40] S. Das, A. Ghosh, and S. S. Mullick, "A switched parameter differential evolution for large scale global optimization—Simpler May be better," in *Proc. MENDEL*, 2015, pp. 103–125.

[41] M. M. Ali and A. A. Törn, "Population set-based global optimization algorithms: Some modifications and numerical studies," *Comput. Oper. Res.*, vol. 31, no. 10, pp. 1703–1725, 2004.

[42] J. Tvrdık, "Competitive differential evolution," in *Proc. MENDEL*, 2006, pp. 7–12.

[43] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE CEC*, 2005, pp. 1785–1791.

[44] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE CEC*, 2008, pp. 1110–1116.

[45] Z. Yang, X. Yao, and J. He, "Making a difference to differential evolution," in *Proc. Adv. Metaheuristics Hard Optim.*, 2008, pp. 397–414.

[46] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *Proc. IEEE CEC*, 2013, pp. 71–78.

[47] R. Tanabe and A. Fukunaga, "TPAM: A simulation-based model for quantitatively analyzing parameter adaptation methods," in *Proc. GECCO*, 2017, pp. 729–736.

[48] Z. Zhao, J. Yang, Z. Hu, and H. Che, "A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric Latin hypercube design for unconstrained optimization problems," *Eur. J. Oper. Res.*, vol. 250, no. 1, pp. 30–45, 2016.

[49] 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.

[50] 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.

[51] T. Takahama and S. Sakai, "Efficient constrained optimization by the ϵ constrained rank-based differential evolution," in *Proc. IEEE CEC*, 2012, pp. 1–8.

[52] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug. 2015.

[53] M. G. H. Omran, A. A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Proc. CIS*, 2005, pp. 192–199.

[54] P. Kaelo and M. M. Ali, "A numerical study of some modified differential evolution algorithms," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 1176–1184, 2006.

[55] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. IEEE CEC*, vol. 3, 1999, pp. 101–106.

[56] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proc. IEEE CEC*, vol. 1, 2001, pp. 94–100.

[57] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, 2005.

[58] M. A. Lee and H. Takagi, "Dynamic control of genetic algorithms using fuzzy logic techniques," in *Proc. ICGA*, 1993, pp. 76–83.

[59] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 673–686, 2006.

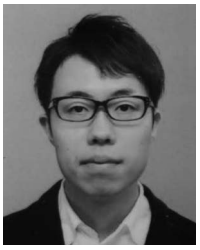
[60] H. A. Abbass, "The self-adaptive Pareto differential evolution algorithm," in *Proc. IEEE CEC*, 2002, pp. 831–836.

[61] H. Guo *et al.*, "Differential evolution improved with self-adaptive control parameters based on simulated annealing," *Swarm Evol. Comput.*, vol. 19, pp. 52–67, Dec. 2014.

[62] Z. Guo and X. Yan, "Optimization of the p-xylene oxidation process by a multi-objective differential evolution algorithm with adaptive parameters co-derived with the population-based incremental learning algorithm," *Eng. Optim.*, vol. 50, no. 4, pp. 716–731, 2018.

[63] 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.

[64] 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, 2011.
[65] N. Noman, D. Bollegala, and H. Iba, "An adaptive differential evolution algorithm," in *Proc. IEEE CEC*, 2011, pp. 2229–2236.
[66] G. Iacca, F. Caraffini, and F. Neri, "Continuous parameter pools in ensemble differential evolution," in *Proc. IEEE SSCI*, 2015, pp. 1529–1536.
[67] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies," in *Proc. SEMCCO*, 2010, pp. 71–78.
[68] G. Dick, "The utility of scale factor adaptation in differential evolution," in *Proc. IEEE CEC*, 2010, pp. 1–8.
[69] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Comput.*, vol. 15, no. 11, pp. 2141–2155, 2011.
[70] J. Cheng, G. Zhang, F. Caraffini, and F. Neri, "Multicriteria adaptive differential evolution for global numerical optimization," *Integr. Comput.-Aided Eng.*, vol. 22, no. 2, pp. 103–107, 2015.
[71] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with discrete mutation control parameters," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1551–1572, 2015.
[72] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 219–232, Jan. 2016.
[73] A. P. Piotrowski and J. J. Napiorkowski, "Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure?" *Swarm Evol. Comput.*, vol. 43, pp. 88–108, Dec. 2018.
[74] P. Pošík and V. Klema, "JADE, an adaptive differential evolution algorithm, benchmarked on the BBOB noiseless testbed," in *Proc. GECCO (Companion)*, 2012, pp. 197–204.
[75] R. Tanabe and A. Fukunaga, "Tuning differential evolution for cheap, medium, and expensive computational budgets," in *Proc. IEEE CEC*, 2015, pp. 2018–2025.
[76] M. Zhabitsky and E. Zhabitskaya, "Asynchronous differential evolution with adaptive correlation matrix," in *Proc. GECCO*, 2013, pp. 455–462.
[77] N. Hansen, A. Auger, D. Brockhoff, D. Tusar, and T. Tusar, "COCO: Performance assessment," *CoRR*, vol. abs/1605.03560, 2016. [Online]. Available: https://dblp.uni-trier.de/rec/bibtex/journals/corr/HansenABTT16
[78] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Mar. 2011.
[79] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.
[80] E. K. Burke *et al.*, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, 2013.
[81] A. Zamuda and J. Brest, "Self-adaptive control parameters' randomization frequency and propagations in differential evolution," *Swarm Evol. Comput.*, vol. 25, pp. 72–99, Dec. 2015.
[82] A. P. Piotrowski, "Review of differential evolution population size," *Swarm Evol. Comput.*, vol. 32, pp. 1–24, Feb. 2017.
[83] 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, Rep. 201311, 2013.



Ryoji Tanabe (M'16) received the Ph.D. degree in science from the University of Tokyo, Tokyo, Japan, in 2016.

He is a Research Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. He was a Post-Doctoral Researcher with ISAS/JAXA, Sagamihara, Japan, from 2016 to 2017. His current research interests include stochastic single-objective and multiobjective optimization algorithms, parameter control in evolutionary algorithms, and automatic algorithm configuration.

Alex Fukunaga received the Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles, CA, USA.

He is a Professor with the Department of General Systems Studies, Graduate School of Arts and Sciences, University of Tokyo, Tokyo, Japan. He was on the faculty of the Tokyo Institute of Technology, Tokyo. He is a Researcher with NASA/Caltech Jet Propulsion Laboratory, Pasadena, CA, USA. His current research interests include heuristic search algorithms, evolutionary computation, and automated planning and scheduling.