Swarm and Evolutionary Computation 43 (2018) 284–311

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy

Rawaa Dawoud Al-Dabbagh [a,*], Ferrante Neri [b], Norisma Idris [c], Mohd Sapiyan Baba [d]

[a] Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq
[b] Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, UK
[c] Department of Artificial Intelligence, Faculty of Computer Science & Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
[d] Computer Science Department, Gulf University for Science and Technology, Kuwait

ARTICLE INFO

Keywords:
Metaheuristic algorithms
Optimization algorithms
Evolutionary algorithms
Genetic algorithm
Parameter control
Differential evolution
Adaptive differential evolution
Review

ABSTRACT

The performance of most metaheuristic algorithms depends on parameters whose settings essentially serve as a key function in determining the quality of the solution and the efficiency of the search. A trend that has emerged recently is to make the algorithm parameters automatically adapt to different problems during optimization, thereby liberating the user from the tedious and time-consuming task of manual setting. These fine-tuning techniques continue to be the object of ongoing research. Differential evolution (DE) is a simple yet powerful population-based metaheuristic. It has demonstrated good convergence, and its principles are easy to understand. DE is very sensitive to its parameter settings and mutation strategy; thus, this study aims to investigate these settings with the diverse versions of adaptive DE algorithms. This study has two main objectives: (1) to present an extension for the original taxonomy of evolutionary algorithms (EAs) parameter settings that has been overlooked by prior research and therefore minimize any confusion that might arise from the former taxonomy and (2) to investigate the various algorithmic design schemes that have been used in the different variants of adaptive DE and convey them in a new classification style. In other words, this study describes in depth the structural analysis and working principle that underlie the promising and recent work in this field, to analyze their advantages and disadvantages and to gain future insights that can further improve these algorithms. Finally, the interpretation of the literature and the comparative analysis of the algorithmic schemes offer several guidelines for designing and implementing adaptive DE algorithms. The proposed design framework provides readers with the main steps required to integrate any proposed meta-algorithm into parameter and/or strategy adaptation schemes.

1. Introduction

Real-world optimization problems are found in a wide variety of applications, such as machine learning, system design and nearly all areas (i.e. disciplines) of science and engineering. The extensive research interest on providing the best solutions to these problems is ongoing. The main challenge such research confronts is often related to uncertainties and/or noise that can lead to theoretical optima, which are by no means optimal or practical in real life [8,60]. Efficient algorithms are known and used to solve typical optimization problems, whereas heuristic methods are used to solve difficult optimization problems [61,62]. However, the increase in the complexity of the problems, in conjunction with the need to decrease the time consumed for thorough problem analysis and tailored algorithm design, implies a demand for robust algorithms with satisfactory performance. These algorithms should not merely adhere to specific problems but are applicable to a wide range of problems and yield good solutions (although unnecessarily optimal) [39,51,111]. Under this class of optimization algorithms are metaheuristic algorithms, such as evolutionary algorithms (EAs) and swarm intelligence algorithms (SIA), which are used to find the parameters (or structures) that maximize or minimize user-defined objective functions. The development of these algorithms continues to be heated and remains an ongoing research task [17,147]. Consequently, algorithmic designers or developers must consider another challenge in addition to problem specifics; it is the well design of the intrinsic part of the meta-algorithm itself [140]. This includes certain algorithm specifics or capabilities, such as maintaining the diversity and mixing among the solutions. Well-designed algorithms must have such capabilities retained for

* Corresponding author.
E-mail addresses: rawaa.dabbagh@ieee.org (R.D. Al-Dabbagh), fneri@dmu.ac.uk (F. Neri), norisma@um.edu.my (N. Idris), sapiyan.m@gust.edu.kw (M.S. Baba).

https://doi.org/10.1016/j.swevo.2018.03.008
Received 14 July 2017; Received in revised form 9 February 2018; Accepted 18 March 2018
Available online 9 April 2018
2210-6502/© 2018 Elsevier B.V. All rights reserved.

balancing global exploration and intensive local exploitation given that these two antagonisms constitute the efficient search ability [48].

The dialects of all metaheuristics are generally based on the same generic framework whose details need to be specified to obtain a particular algorithm. These details are customarily called algorithm parameters, such as probability of mutation, probability of crossover, tournament size of selection and size of population. Many studies on EAs examine the implementation of adaptive EAs [29,76]. This type of EAs, if well designed, can enhance the robustness and convergence performance of the algorithm by dynamically updating the EA parameters for different objective function landscapes during evolution [63]. This has led to the on-the-fly alteration for such parameters during evolution by accounting for the actual search progress to achieve optimal convergence and liberate a user from the tedious trials of tuning the parameters. As discussed in Refs. [38] and [39], the main idea is to no longer choose the parameters in a semi-arbitrary fashion[1] but to allow the parameters to adapt themselves to the problem.

More than a decade ago, differential evolution (DE) algorithm emerged as a special and competitive form of metaheuristics [35]. Owing to its distinct features, DE algorithm can be viewed from two different perspectives. It can be classified as EA because of its recombination and selection operators, in addition to the names and descriptions of its control parameters. At the same time, DE algorithm can also be regarded as SIA because of its structure that, over a number of generations, a group of initiated particles tend to converge their values closer to the particle whose value is closest to the target at any given moment [96,138]. Notwithstanding of its classification, DE algorithm and its numerous variants have developed rapidly as simple and robust algorithms. Practitioners from different disciplines of science and engineering have applied DE algorithms to address various optimization problems in their own fields, regardless of whether these problems are continuous, combinatorial or mixed variable [2,3]. DE has produced superior results across widely used benchmark functions [41,136] and real-world applications [55,150]. Table 1 covers excerpts of the most prominent milestones and epochs in the DE history starting from the time it was proposed by Storn and Price in 1995 [118,120] until present. The events in Table 1 are presented chronologically. This arrangement can be observed from the years that correspond to each subject (line). However, certain topics require citations from past to recent years. Consequently, two markers have been used in the Juncture column. The first marker is a small circle that refers to a specific work or publication in the DE literature. The second marker is a line that refers to an unspecified number of publications in the DE literature that are related to one topic starting from the year during which the first work regarding that topic was published. For a subject with a line marker, related and recent references are cited as examples.

Claims and counterclaims have already been proposed, especially by engineers, regarding the rules in choosing the appropriate control values of standard DE parameters with which to solve practical problems [45, 154]. However, the performance of DE algorithm depends heavily on the selected mutation strategy and its associated control parameters. The sensitivity of the DE algorithm to its mutation strategy and its corresponding control parameters can significantly deteriorate its performance if the strategy is improperly selected [4,133]. Choosing a suitable DE strategy and setting its control parameters is difficult and requires much user experience. In the past few years, many researchers have attempted to make the algorithm into a general and fast optimization method for any kind of optimization problem by tuning its various constitutes, such as initialization, mutation, diversity enhancement, and crossover of DE [9,34]. Multiple attempts have also been conducted to automatically adjust the algorithm's parameters for single or multiple problems. The development of (self-)adaptive DE algorithms has resulted in faster and more reliable convergence performance in many benchmark problems than classical DE algorithms with manual parameter settings [18,149,153].

The present study aims to provide insights for fresh and experienced practitioners alike who are interested in the field of parameter settings in metaheuristics. This study is designed from the ground up to support the issue of controlling the values of various parameters of an evolutionary algorithm in general and of DE algorithm in particular. The main objective of this study is to present a comprehensive procedural analysis (in Section 5) that is conducted to investigate the various adaptive schemes utilized to automatically control the DE parameters and/or its mutation strategies. For this purpose, two taxonomies are proposed in this study. The first taxonomy (in Section 3-Fig. 2) is proposed to eliminate any ambiguity related to classify any adaptive EA. The new classification comprises three levels of categories instead of two regarding the parameter control type (deterministic, adaptive, self-adaptive) and the evidence (absolute, relative) used for determining the change of the parameter. The second taxonomy (in Section 5-Fig. 3) is a new taxonomy proposed to classify the adaptive DE algorithms into two categories (DE with adaptive parameters and DE with adaptive parameters and mutation strategies). This study comprehensively describes the structural analysis and working principle that underlie the promising work of these algorithms. The study also discusses the advantages and disadvantages of these algorithms and suggests future insights that can further improve their algorithmic design. Eventually, protocols for future adaptive DE implementations are also offered (in Sections 6-7).

The rest of this paper is organized as follows. Section 2 provides an overview of the published survey work on adaptive DE algorithms. Section 3 presents an extended taxonomy of EA parameter settings. Section 4 discusses the standard procedural design of DE algorithm. Section 5 applies the two proposed taxonomies (in Figs. 2 and 3) to multiple adaptive DE algorithms specifically as an example to convey the main purpose of these taxonomies. A procedural analysis is then established on these algorithms to elucidate the conceptual similarities and differences among them and the pros and cons of each adaptive scheme. Section 6 presents a general framework that lists the steps that should be considered to create a meta-algorithm with parameter control. Section 7 concludes the paper and summarizes the objectives addressed. Suggestions and future work developments are also offered in this section.

2. Related state-of-the-art work: in what way is this survey different to the others?

Since numerous DE variants have been proposed in the literature, many surveys on DE have been published. Several survey papers have fully captured the performance aspects of adaptive and self-adaptive DE. Das and Suganthan [36] published a comprehensive survey that addressed nearly all scenarios where DE is applied, such as DE and constrained optimization functions, important DE schemes for single-objective functions, DE in complex environments, theoretical analysis development of DE and most contemporary engineering applications of DE. The review of Das and Suganthan [36] also briefly introduced the control of DE parameters and presented the most prominent and recent DE variants in the field. The survey of Neri and Tirronen [93] presented DE algorithms and their most recent advances in a classification format. The reviewed algorithms were categorized into two classes. The first class is based on integrating DE with an additional component, such as local search methods. The second class is based on modifying the DE structure, such as algorithms with adaptive schemes. The study also conducted detailed experiments based on a wide set of benchmark problems to test the overall performance of these algorithmic classes.

Chiang et al. [28] published a new taxonomy for DE parameter control mechanisms based on the type of parameter values (discrete and continuous), number of parameter values (multiple, individual and variable) and the information used to adjust the parameter values (random, population, parent and individual).

Suganthan et al. [123] published a review paper on adaptive DE

[1] The choices were often based on experience.

Table 1
Historical elucidation of the differential evolution algorithm inventions and developments [1,2,5,16,21,22,26,34–37,41–44,46,47,55,56,65–68,71,72,75,77,80–84,87,88,90–95,100–102,107,109,114,116–120,123,126,129–134,137,138,141–145,152,153].

Subject	Citation (publication)	Juncture
The first technical report was written on DE.	[118]	
The successful of DE was demonstrated at the First International Contest on Evolutionary Optimization (1st ICEO).	[119]	
Two journal articles published describing DE in detail and its very good results	[100], [120]	
DE was presented at the Second International Contest on Evolutionary Optimization (2nd ICEO)	[101]	
A compendium on DE 'New Ideas on Optimization' has been summarized by Price.	[102]	
Preliminary recommendations on how to choose appropriate parameter settings of DE.	[68], [100], [101], [102], [116]	
The first modifications of DE selection rule for constraints handling, were presented.	[66], [67], [91]	
New mutation schemes were presented and some other developed DE strategies.	[42], [46], [47], [83], [87]	
Two types of crossover schemes were considered: binomial and exponential. In 2011, two new crossover schemes were designed: consecutive binomial crossover and non-consecutive exponential crossover.	[72], [102], [144]	
Integration based randomization of individuals with explicit exploitative component: Memetic differential evolution and its variants.	[92], [94]	
Structure based randomization of individuals: Compact DE.	[82]	
Increase the search moves of DE with structured population: Parallel DE; Distributed DE; Micro-DE	[26], [137], [138] more about parallel DE in [35]	
The era of developing adaptive DE started in 2002 and continues to be a very hectic trend until current time (will be discussed in details in SECTION 5).	[1], [21], [34], [43], [123], [126], [129], [131], [132], [133], [142], [152], [153]	

Juncture: 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017

Specified number of DE publications | Unspecified number of DE publications | Unspecified number of DE publications in the field of parameter settings

Subject		Citation (papers)
DE in a wide range of real-world problems.	Many applications of DE to engineering problems until 2015 have been summarized in two surveys.	[35], [36]
	Optimization of multidimensional discrete point clouds (PCs).	[130]
	Maximization of trading profit in financial market.	[90]
	Photonic structure designs.	[16]
	High-precision gear design.	[2]
	In communication industry, e.g. optimization of power consumption of Long Term Evolution (4G LTE) base stations.	[55], [117]
	Fast 3D path planning.	[141]
Significant improvements have been presented in many aspects in the overall framework of DE which are still ongoing.	DE in constrained optimization.	[37], [41], [65], [80], [84], [88], [145]
	DE in multi-objective optimization.	[5], [75], [107], [109], [114], [134]
	DE for global optimization in dynamic environments.	[22], [34], [56], [95]
	Multi-search or multi-population DE.	[71], [77], [95], [143]
	DE in Multifactorial Optimization (MFO)	[44]
Surveys that cover many published aspects of DE (till 2015).		[35], [36], [81], [93], [123]

Juncture: 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017

Specified number of DE publications | Unspecified number of DE publications | Unspecified number of DE publications in the field of parameter settings

variants. In their review, the authors introduced a simple classification of adaptive DE algorithms according to the type of optimization problem that they are dealing with: adaptive DE for a single optimization problem, adaptive DE for other problem scenarios and adaptive population topology in DE. Several adaptive DE algorithms, such as DE with a self-adaptive trial vector generator [20] and DE with an adaptive trial vector generator and parameter control [104] have been listed under the three proposed classes. Although several adaptive DE algorithms with applications in multiple problem scenarios were included in this literature survey, no general adaptive classification that covered the different aspects of published adaptive DE was proposed. Moreover, no thorough procedural analysis with the pros and cons of these algorithms was provided. Lastly, this review is obsolete.

Das et al. [35] published an updated version of Das and Suganthan's

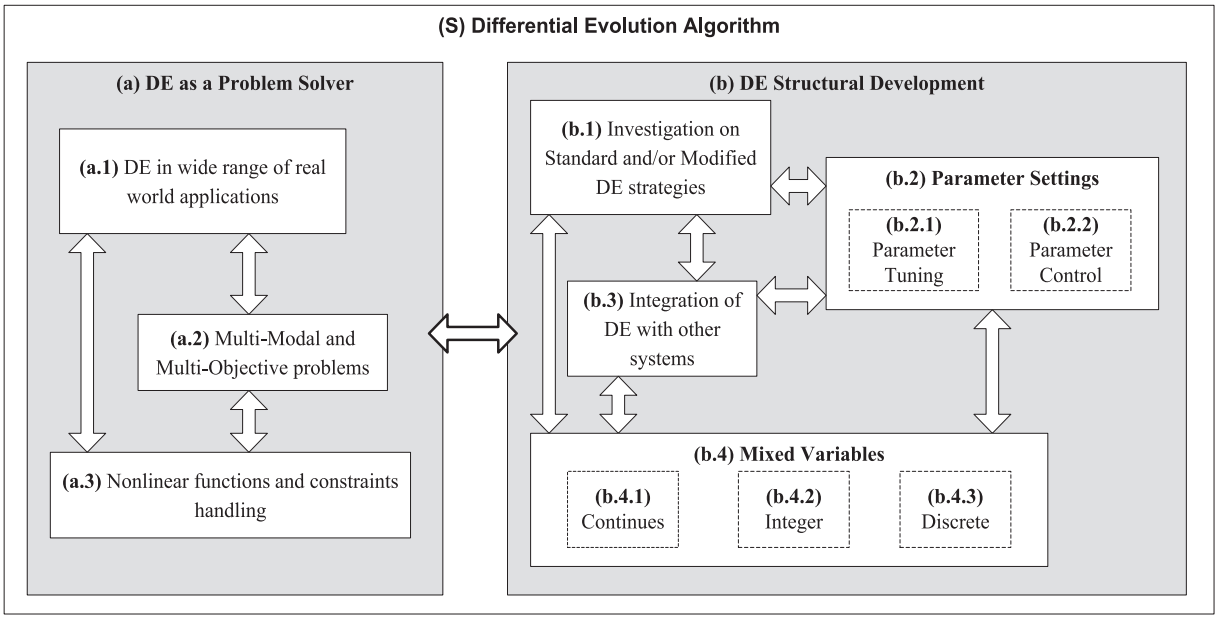

Fig. 1. Combination of differential evolution state-of-the-art flow diagram.

[36] survey paper. In their work, a complete section was devoted to cover recent adaptive DE algorithms in a classification format, such as DE with adaptive scalar factor (F) and crossover rate (CR) parameters only, DE with both adaptive strategy and control parameters (F and CR) and DE with adaptive population size (Np). However, we wish to raise some points about this section of the survey.

- Algorithms included under this classification are dated between 2011 and 2015. New versions of L-SHADE, such as LSHADE-*cn*EpSin [10], have been published recently. We have referred to these versions in our study and other new versions of adaptive DEs have also been cited and discussed.
- We have found algorithms with special adaptive properties that have been included in classes with different adaptive merits. For example, Sarker et al. [112] and Tanabe and Fukunaga [125] are included in the class where F and CR are the only adaptive parameters; however, these algorithms also update the value of Np. Similarly, Zamuda et al. [155] is included in the class where F and CR are the only adaptive parameters; however, this work also adapts the Np value and the trial vector generation schemes. Finally, Brest and Maucec [21], Zamuda and Brest [156] and Zamuda et al. [155] are included in the class of adaptive DE with Np; however, these algorithms also exhibit the adaptation property of multiple mutation strategies, which can be overlooked by readers.
- The discussion about the impact of the type of selected mutation strategy and parameter adaptation scheme is nearly missing. The analysis about the pros and cons of each algorithm (such as in Section 5) and future work that can be conducted to extend the referenced studies (such as in Section 7) are also missing.
- We have thoroughly discussed a critical issue regarding adaptive algorithms in our paper, which has been overlooked by many survey studies on this topic. The issue being referred to is the classification of parameter control (and its extended version) in any metaheuristic approach (Section 3). This classification is proposed to eliminate any ambiguity related to the classification and understanding of the adaptive characteristics of any adaptive metaheuristic and to complete any survey related to adaptive algorithms. For example, an adaptive probability-based scheme was used to adapt mutation strategies in Ref. [24] according to the 2016 survey of Das et al. By contrast, a deterministic parameter control scheme was used to select the values of F and CR for each strategy during the run, which makes this algorithm different from others. These details concerning adaptive algorithms must be highlighted.

Moreover, a survey paper typically attracts new researchers in the field. Thus, sufficient background about the topic it addresses should be provided.

Finally, the main objectives of these review papers are to report, classify and categorize DE versions presented in the literature. Unlike them, the current article focuses on adaptation with the purpose to perform the algorithmic design of DE algorithms. The interpretation of the literature and the comparative analysis of the results offer several guidelines on how to design and implement adaptive DE algorithms (Section 6). The proposed designing framework provides readers with the main steps required to integrate any proposed meta-algorithm into parameter and/or strategy adaptation schemes. Moreover, we design a state-of-the-art DE schematic flow by customizing a distinct alphabetical index for each contribution aspect considering the strong and multifaceted contribution trends of the DE algorithm and our tendency towards simplicity, as depicted in Fig. 1. As shown in the figure, some or even all areas of a DE algorithm can overlap in a common work. We eventually found that surveys that deal with the control of DE parameters (Fig. 1-b.2) are rare or obsolete [28,35]. This research gap has encouraged us to conduct the current comprehensive survey to cover certain aspects related to DE parameter control.

3. Metaheuristics parameter settings: extended taxonomy

The critical decision in implementing any meta-algorithm is on how to set the values for various parameters of that algorithm. These values greatly affect the performance of the evolution. Parameter settings are commonly composed of crossover rate, mutation step, population size, selection pressure, and penalty coefficient. It is an important and promising aspect of metaheuristics. As an instance, the efficiency of an EA greatly depends on the setting of these parameters, that is, by *parameter tuning* or *parameter control* [4,63]. Parameter tuning is also called off-line setting and involves using several “standard” parameter setting values in advance and keeping these settings fixed during the run, whereas parameter control is also called *on-line setting* and involves using another class of approaches where parameters are subject to *change* or *evolve* as problem parameters are optimized. Scientists and practitioners typically tune EA parameters manually and are guided only by their experience

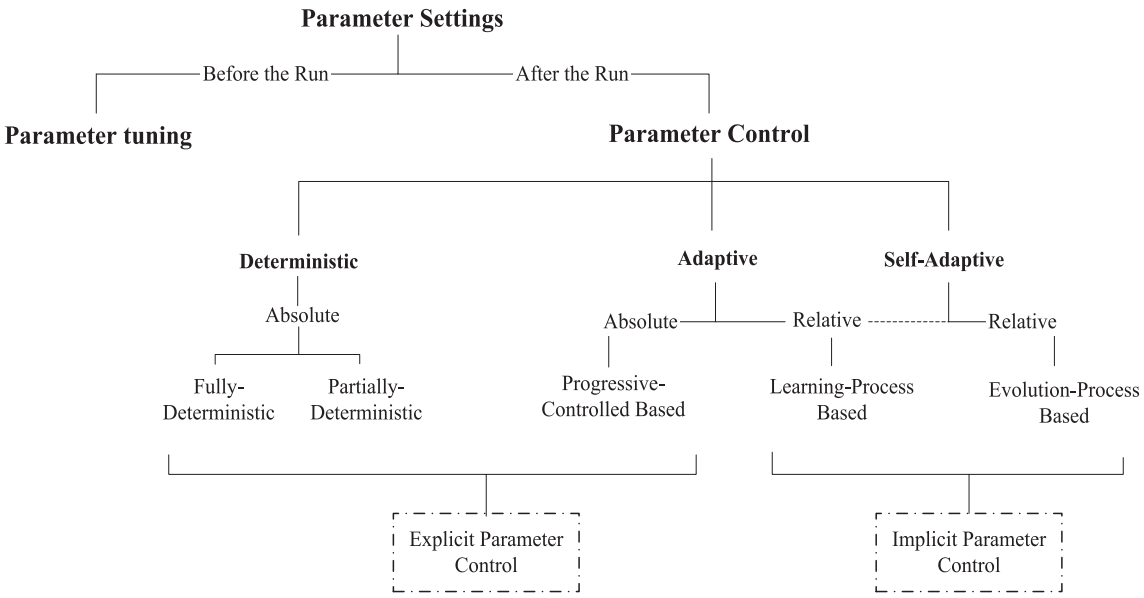

Fig. 2. Extended taxonomy of parameters settings in EAs.

and some rules of thumb. Parameter tuning often requires tedious and time-consuming human involvement. Moreover, the process of any EA is essentially adaptive and dynamic process; thus, using fixed parameters with constant values opposes this essence. Intuitively, the values of these parameters might be optimal at different stages of the evolution; any effort spent toward this direction is indeed lost a priori [7,18,29,38,39]. The downsides or limitations of parameter tuning are as follows [76]:

- Parameter values tuned for a single problem may lead to a large difference in performance if these parameters were set to different values.
- A parameter tuned for one test problem and produced superior results may not be as effective in other problems.
- EA parameters are intrinsically dependent; thus, tuning them independently is inconvenient.

An alternative form is parameter control, which refers to when an automated setting is applied on EA parameter values. Globally, the automation of parameter settings encompasses three main categories [29,39,76]:

- *Deterministic parameter control* – automation occurs when a deterministic rule is triggered to modify the value of a strategy parameter in a fixed, predetermined manner without using any feedback from the search.
- *Adaptive parameter control* – automation occurs during the evolution when the strategy parameter direction and/or magnitude are adjusted according to a pre-designed rule. Basically, automation incorporates information gleaned from the feedback based on algorithm performance, such as the quality of the individual fitness value, without being part of the evolution, where the new control parameter value may or may not persists or propagates throughout the next iterations.
- *Self-adaptive parameter control* – automation occurs when the strategy parameters undergo genetic encoding and when the alteration is subject to evolution and pressure (i.e., mutation and crossover); better parameter values tend to produce better individuals (i.e., solutions) with the highest chance to survive and propagate for more off-springs.

Another important criterion that should be considered when discussing parameter control techniques is the evidence of change in parameter value, which can be observed from the performance of operators, the diversity of the population, and fitness values. Evidence can be absolute or relative. Absolute evidence is when a rule is applied to alter a strategy parameter value on the basis of a predefined event feedback, such as updating the probability of mutation rate in accordance with a fuzzy rule set, population diversity drops at some given value, and even time elapses, rather than being relative to the performance of other values. By contrast, relative evidence is when the strategy parameter value is altered according to the fitness of the offspring produced and the better is rewarded; this change is specified relative, not deterministically, to one value present at any time. Therefore, deterministic parameter control is impossible with relative evidence and thus for self-adaptive parameter control with absolute evidence [7,29,38,39].

The aforementioned terminologies of the parameter setting of EAs have led to the taxonomy illustrated in Fig. 2. The new taxonomy is an extension of a former one suggested in Ref. [39] which caused some confusion among a number of researchers working in this field, particularly in distinguishing deterministic and absolute adaptive rule, as well as relative adaptive rule and self-adaptive rule.

Accordingly, we investigated the subject of parameter control and found new subcategories that can be added to the main one to address the ambiguity in classification. The definitions of these subcategories are as follows:

- *Fully-Deterministic Scheme and Partially-Deterministic Scheme*, these two subcategories fall under *Deterministic Parameter Control* category. Their main feature is not receiving any feedback from the search during the evolution. Technically, a *fully-predetermined* rule is when a user makes a complete counter-intuition on how to steer the control parameter to the desired direction and/or magnitude; for instance, a rule is triggered on the basis of a certain number of generations that elapsed. By contrast, a *partially-predetermined* rule is when one uses random based scheme to alter, for example, mutation probability after every 100 generations [49].
- *Progressive-Controlled Based,* this subcategory is applied when some feedback from the search is discriminated as measurements on the basis of user pre-determined rules. When these measurements achieve a threshold, a corresponding adaptive rule is applied to update its relative parameter control. Thus, the progress of updating parameter values, as well as the search, is controlled under inconsiderable intuition. For instance, these measurements may be based on gathering information from previous runs through data mining-based

fuzzy-knowledge control [74], theoretical considerations [115], or practical experience encapsulation [73]. A prominent example of this type of parameter control is the 1/5 success rule of Rechenberg [108, 113]; this rule is applied at certain periodic intervals deterministically.

- *Progressive-Uncontrolled Based (Learning Process-Based)*, this subcategory is the most prominent in literature. This sub-classification falls between adaptive rule with relative evidence and self-adaptive with evolution-process rule, because both rules are intersected on the basis of updating the control parameter values associated with each individual, at each generation based on their corresponding fitness value; better features of individuals will be propagated to the next populations over time. The only difference is that in *progressive-uncontrolled rule* feedback is gained from the search to allow the parameter control to gradually adapt by applying a pre-specified strategy, which is most likely analogue to that of crossover and mutation strategies [57,105,149]; most of these strategies are learning schemes that collect experience from previous search. In such methods, the changes performed on the parameter values are fully uncontrolled, because the adaptive rule is associated only with the "fittest" solutions and its corresponding control parameters. This subcategory causes much confusion for some researchers working in this field [149]. For convenience, we use dashed-line connector to make it optional for researchers who desire to stick with the former taxonomy, not to use the latter one, and include the learning style under self-adaptive category, as shown in Fig. 2.

Categories that are involved in the search and gradually evolved by either learning or evolution strategy as long as the search is not yet terminated, are considered as implicit parameter control, otherwise, are explicit parameter control.

Ultimately, recent studies have examined the inclusion of algorithms' operators under the new extended classification (i.e. Fig. 2) as well to yield new algorithms with adaptive parameters and operators. These algorithms have shown substantial performance improvements compared with algorithms with adaptive parameters only [58,98]. Accordingly, self-adaptive algorithms are defined as algorithms with operators and parameters encoded together with their associated solutions and evolved during the run, whereas adaptive algorithms are defined as separate adaptation-based performance algorithms in which parameters and operators that yield good solutions are rewarded and propagated to the next generations [122], as discussed in detail in Section 5.

4. Differential evolution: standard structure

Differential evolution (DE) is a robust and competitive metaheuristic method for real parameter optimization [118]. Its effectiveness in solving a wide variety of optimization problems in science and engineering domains has been proven [36]. This effectiveness is due to its simple concept, which makes it easy to implement. Classical DE features few control parameters (scalar factor F, crossover rate CR and population size Np), although they greatly affect the performance of DE [126,149] which will be discussed in the next section. After the initialization stage, DE enters a flow cycle of stages (mutation, crossover and selection) which is repeated in that order for G_{max} generations.

4.1. DE vector initialization

DE, like many EAs, is a population-based optimizer that starts attacking the problem by generating multiple, uniformly and randomly chosen points as initial samples to create population P. Usually, each point x of P represents a vector of D-dimension in a real parameter space $\mathbb{R}^D$ like $x = [x_1, x_2, \ldots, x_D]$ and is indexed with a number between 1 and Np. In the first generation ($G = 0$), these vectors are initiated according to lower $x_{min} = [x_{1,min}, x_{2,min}, \ldots, x_{D,min}]$ and upper $x_{max} = [x_{1,max}, x_{2,max}, \ldots, x_{D,max}]$ bounds of each decision variable (problem parameter) x_j of x as

$$x_{j,i}^{G=0} = x_{j,min} + rand[0,1]_{j,i} \cdot \left(x_{j,max} - x_{j,min}\right) \tag{1}$$

where $rand[0,1]$ is a uniformly distributed random number generated between 0 and 1.

In DE community, the vectors generated at each stage of DE cycle have different names. For example, the vectors or population created at the beginning of each generation are called *target vectors/target population* and denoted as x.

4.2. DE mutation operation

In the preceding stage of generating the target vectors $x_i (i = 1, 2, \ldots, Np)$, mutation operation is performed to generate *donor vectors/donor population* $v_i (i = 1, 2, \ldots, Np)$. The main idea of this operation (see Eq. (2)) is to add a scaled difference of two target vectors (x_{r1} and x_{r2}) to a third vector x_{r3} from the current population to yield the mutant vector v_i. The three indices ($r1$, $r2$ and $r3$) are mutually distinct, chosen from the range $[1, 2, \ldots, Np]$, and do not coincide with the index i of the current vector. This operation is also called *differential mutation.*

$$v_i^G = x_{r1}^G + F \cdot \left(x_{r2}^G - x_{r3}^G\right) \tag{2}$$

In Eq. (2), G represents the current generation, and F is a control parameter that determines the exploration length $(x_{r2} - x_{r3})$ governing how far the offspring from point x_{r1} should be generated. $F \in [0, 1+[$ always takes positive value which cannot be greater than 1 [103]. DE has many nomenclatures in the literature derived from its mutation formulas. The DE scheme shown in Eq. (2) is called DE/rand/1. Other DE schemes are expressed below.

$$\text{DE/best/1} \quad v_i^G = x_{best}^G + F \cdot \left(x_{r1}^G - x_{r2}^G\right) \tag{3}$$

$$\text{DE/current} - \text{to} - \text{best/1} \quad v_i^G = x_i^G + F \cdot \left(x_{best}^G - x_i^G\right) + F \cdot \left(x_{r1}^G - x_{r2}^G\right) \tag{4}$$

$$\text{DE/rand/2} \quad v_i^G = x_{r1}^G + F \cdot \left(x_{r2}^G - x_{r3}^G\right) + F \cdot \left(x_{r4}^G - x_{r5}^G\right) \tag{5}$$

$$\text{DE/best/2} \quad v_i^G = x_{best}^G + F \cdot \left(x_{r1}^G - x_{r2}^G\right) + F \cdot \left(x_{r3}^G - x_{r4}^G\right) \tag{6}$$

where x_{best} is the best solution found so far in the current generation G.

4.3. DE crossover operation

The next stage that follows mutation in the DE flow cycle is crossover or vector perturbation operation to introduce the *trial vectors/trial population* $u_i (i = 1, 2, \ldots, Np)$. This operation is responsible for increasing the diversity among the parameters of target x_i^G and v_i^G donor vectors. Under this operation, the target and donor vectors exchange their components with respect to predefined conditions to form the trial vector u_i^G. DE family has two types of crossover −*binary* (*uniform*) crossover and *exponential* crossover. In what follows, the binary crossover is defined, which is also known as *bin.*

$$u_{j,i}^G = \begin{cases} v_{j,i}^G & \text{if } rand_{j,i}[0,1] \le CR \quad or \quad j = j_{rand} \\ x_{j,i}^G & otherwise \end{cases} \tag{7}$$

where $rand_{j,i}[0,1]$ is a uniform random number generated between 0 and 1; CR is a DE control parameter just like F. Its value determines the variation probability among the perturbed vectors. $CR \in [0, 1+[$ always takes positive value which cannot be greater than 1 [103]. $j_{rand} \in [1, 2, \ldots, D]$ is the index of a randomly selected gene to ensure that the

offspring u_i^G inherits at least one component from the parent v_i^G.

4.4. DE evaluation and selection operations

In the final stage of DE, a greedy selection scheme takes place (see Eq. (8)) to measure how far has been achieved for the best performance. This scheme is done by comparing the quality of the solution obtained from the crossover stage (trial vector u_i^G) with the quality of its corresponding target vector x_i^G to determine who will survive to the next generation, $x_i^{G+1}(i = 1, 2, \ldots, Np)$. The quality of solution is calculated using the objective function f (fitness function) designed for the problem in hand. This operation is described as

$$x_i^{G+1} = \begin{cases} u_i^G & if\ f(u_i^G) \le f(x_i^G) \\ x_i^G & otherwise \end{cases} \quad \text{(for minimization problems)} \qquad (8)$$

From the above equation, if the trial vector has a lower or equal fitness function value, then it replaces the corresponding target point in the next generation; otherwise, the old point is retained in the next generation. This strategy guarantees that through generations the population will never deteriorate because the population either gets better (with respect to minimizing the fitness value) or remains the same.

5. Adaptive differential evolution: procedural analysis and comparison

Generally, DE disposes three control parameters.

- Population size (Np): The total number of potential solutions in one generation
- Mutation factor (F): The amount of differentiation ratios that the perturbed solution can acquire
- Crossover rate (CR): The probability in which the offspring inherits the actual genes of a parent individual

The literature reveals many recent and prominent adaptive DE variants that show efficiency and reliability in their performance. In this study, notable adaptive DE algorithms (as listed in Fig. 3, there are 28 adaptive DE variants) have been reviewed and analyzed on a case-by-case basis according to a particular situation implemented (i.e. parameter control scheme and/or adaptive DE mutation strategy) within the new proposed taxonomy. Due to space limitation, 19 algorithms out of 28 have been thoroughly presented and discussed; the other 9 algorithms have been summarized in Table 8. These 19 algorithms are as follows:

- FADE (Fuzzy Adaptive DE) is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using fuzzy logic [74].
- jDE is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using self-adaptive scheme [20].
- FiADE (Fitness-Adaptive DE) is a parameter adaptive DE that updates the values of F and CR during evolution based on the fitness function values of the individuals in the population [52].
- GADE (Greedy Adaptive Differential Evolution) is an adaptive DE algorithm with an effective greedy strategy that updates the values of F and CR during the run in a successive learning period [70].
- DESAP is a parameter adaptive DE in which the control parameters (F, CR and Np) of DE are all adjusted though evolution [126].
- JADE is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using self-adaptive learning scheme [149].
- DEGL (DE with global and local neighborhoods) is a parameter adaptive DE with a novel mutation strategy. In this algorithm, new parameter control schemes that balance between the exploitation and exploration abilities in the DE mutation strategy during evolution is presented [31].
- MDE_pBX is a parameter adaptive DE in which the control parameters (F and CR) of DE are adjusted using self-adaptive learning scheme [59].

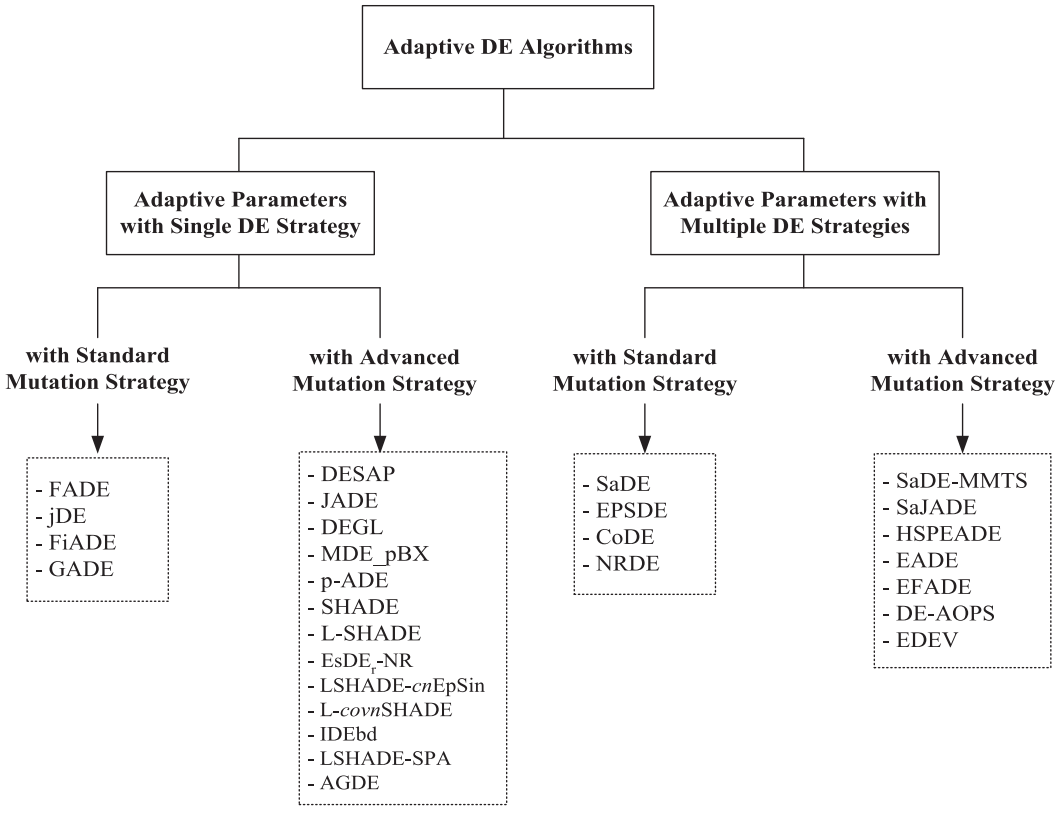

Fig. 3. New classification illustrates the position of each adaptive DE variant with respect to the type of adaptive scheme(s) it applies.

- *p*-ADE is a parameter adaptive DE that adjusts the parameters of *F* and *CR* and other control parameters related to its mutation scheme in an adaptive manner [15].
- SHADE (Success-History based Adaptive DE) is an improved version of JADE algorithm. It proposes an adaptation technique that controls the parameter settings of *F* and *CR* based on a historical memory of successful solutions and it is updated dynamically during the run. The advantage of the historical memory is to keep track of the control parameters values that generate the best solutions [124].
- L-SHADE is an extended version of SHADE algorithm in which a linear population size reduction (LPSR) is integrated to continually decrease the size of DE population using a linear function. This strategy has exhibited efficiency in saving the computation cost of the origin algorithm [125].
- EsDE$_r$-NR (Ensemble sinusoidal differential evolution with niching reduction) is an enhanced version of the LSHADE-EpSin algorithm (L-SHADE with ensemble parameter sinusoidal adaptation) presented in Ref. [11]. EsDE$_r$-NR features many significant adaptation merits in which two sinusoidal approaches and a Cauchy distribution are used as mixture to update the value of *F* during evolution. In addition, the population size *Np* has been reduced during generations using a novel niching-based reduction scheme. Finally, a restart method is activated at the late stage of evolution to further improve the quality of the solutions found so far [9].
- SaDE is a parameter and strategy adaptive DE in which the control parameters (*F* and *CR*) of DE as well as the DE strategies are adjusted using adaptive techniques [104].
- EPSDE is a new version of adaptive DE in which an ensemble of control parameters and strategies are created then selected randomly for each individual [79].
- NRDE (Noise Resilient DE) is an improved DE algorithm with strategy and parameter control proposed for complex optimization problems, mainly, for functions corrupted with additive noise. This algorithm proposes three new mechanisms to achieve this objective: adaptive switching between two alternative mutation strategies, blending crossover, and threshold-based selection mechanism [53].
- SaDE-MMTS is a parameter and strategy adaptive DE in which the control parameters (*F* and *CR*) of DE as well as the DE strategies are adjusted using adaptive techniques. This algorithm is an integration of SaDE, JADE and local search algorithms [151].
- SaM (SaJADE) is a parameter and strategy adaptive DE in which the control parameters (*F* and *CR*) of DE as well as the JADE strategies are adjusted using adaptive techniques. This algorithm is an improvement of the JADE [54].
- EADE is an enhanced adaptive differential evolution algorithm that presents a novel mutation strategy and a new adaptive learning scheme to update the value of *CR* during evolution. Both the mutation strategy and the adaptive scheme have successfully create the balance between the exploration and exploitation capabilities of DE and prove effectiveness in solving large-scale optimization problems [85].
- EFADE is an enhanced fitness-adaptive differential evolution algorithm with novel mutation. In this algorithm, a new triangular mutation strategy is presented to improve the exploration and exploitation capabilities of DE. In addition, two novel adaptation schemes for controlling the values of *F* and *CR* during evolution are suggested [89].

To convey the aforementioned information, this section is divided into THREE major subsections.

5.1. DE with adaptive parameters and single strategy

In this subsection, the main characteristics and mechanisms of 12 remarkable adaptive DE versions are stated in details on the basis of parameter adaptive schemes and DE strategies.

5.1.1. Adaptive DE with single standard strategy

5.1.1.1. FADE algorithm. FADE uses the standard DE scheme DE/rand/1/bin. It updates the values of *F* and *CR* in each generation using a mechanism, which is based on the fuzzy logic controller (FLC), whereby a fuzzy knowledge-based system is used to update the control parameters online in a dynamic adaptive manner to the inconsistent situation.

- The function values, population diversity (FC), parameter vectors (PC), and their updates after n^{th} generations are calculated and then used as input to the FLCs. The values of the control parameters (i.e. *F* and *CR*) are the outputs.
- The values of *F* and *CR* are then assigned to the fuzzy sets membership functions.
- '9 × 2' IF-THEN fuzzy rules statements are used to formulate the conditional statements that comprise fuzzy logic.
- Mamdani's fuzzy inference method is used as the fuzzy control strategy to map from the given inputs to an output.
- Defuzzification is done to map from a space of fuzzy output to a space of real output.

5.1.1.2. jDE algorithm. jDE uses the standard DE strategy DE/rand/1/bin. It updates the values of *F* and *CR* in a self-adaptive manner based on adjusting the control parameters *F* and *CR* by means of evolution and applied at the individual level. First, each individual $x_i^{G=0}(\, i = 1, 2, \ldots, Np\,)$ is associated with its corresponding control parameters F_i and CR_i. In the first generation, these parameters are initialized to $F_i^{G=0} = 0.5$ and $CR_i^{G=0} = 0.9$, then F_i^{G+1} and CR_i^{G+1} are assigned to values generated according to uniform distributions in $[0.1, 1]$ and $[0, 1]$ respectively during evolution as follows:

$$F_i^{G+1} = \begin{cases} 0.1 + rand_1 \times 0.9, & if\ \ rand_2 < \tau_1 \\ F_i^G, & otherwise \end{cases} \tag{9}$$

$$CR_i^{G+1} = \begin{cases} rand_3, & if\ rand_4 < \tau_2 \\ CR_i^G, & otherwise \end{cases} \tag{10}$$

where $rand_j$ and $j \in \{1, 2, 3, 4\}$ are uniform random values $\in [0, 1]$. In this algorithm, τ_1 and τ_2 represent the probability limits that permit the adjustment of *F* and *CR* values, and they are both assigned to the same value 0.1.

A recent study has been devoted to present a thorough statistical analysis to show the influence of tuning the randomness level parameter in the performance of jDE-like algorithms. The study has suggested values for the control parameters and the means to select them to better steer the iteration-temporal performance of the algorithms. Many open issues for the applications of the self-adaptive DE mechanisms into other mechanisms have also been indicated such as, in the online learning of ensemble strategies [146].

5.1.1.3. FiADE. In this algorithm, novel adaptation schemes for adjusting the values of *F* and *CR* online are proposed and then integrated with the standard DE/best/1/bin algorithm. The underlying idea behind these adaptive schemes is the best individual in the current generation will suffer from decreasing its step-size at the mutation stage to increase its chances of searching nearby the optimum area (neighborhood). At the crossover stage, the same individual will have its components propagated to its offspring more than the other individuals in the same population. Using these strategies creates an appropriate balance between the exploitation and exploration abilities of DE. FiADE is implemented with regard to the fitness function values of the latest individuals obtained, and this is the second attempt after [6] who proposed a fitness-based adaptation scheme for *F*. Firstly, in FiADE, the following two different schemes (Scheme 1 and Scheme 2) are used to adjust the value of *F* during evolution.

$$\text{Scheme1:} \quad F_i = F_{max} \cdot \left(\frac{\Delta f_i}{\lambda + \Delta f_i} \right) \tag{11}$$

where $\Delta f_i = |f(x_i) - f(x_{best})|$ and $\lambda = \frac{\Delta f_i}{10} + 10^{-4}$. Number 10^{-4} is added to avoid the problem of division by zero when $\Delta f_i = 0$.

$$\text{Scheme 2:} \quad F_i = F_{max} \cdot \left(1 - e^{-\Delta f_i}\right) \tag{12}$$

In both schemes, $F_{max} = 0.8$ because this setting gives best results over various benchmark functions through experiments. The two equations above clearly show that when $\Delta f_i \to 0$ then $F_i \to 0$. For scheme 1, $F_i \to 0.8$ when Δf_i is large and $\lambda / \Delta f_i \to 0$. Likewise, in scheme 2, $F_i \to 0.8$ when Δf_i is large and $e^{\Delta f_i} \to 0$.

These two schemes play a key role in satisfying the objective of generating the best values of F during evolution in a consecutive manner. Whenever the value of Δf is greater than a threshold value (in the original paper [52] it is 2.4), scheme 2 will always have a greater value of F to keep the exploration process active. On the contrary, when Δf drops less than the value of the threshold, the search moves towards premature convergence as the value of F decreases drastically. At this stage, scheme 1 is used to solve the degradation of the evolution as it decreases the value of F at a lower rate, thus preventing the algorithm from falling into premature termination. Therefore, the adaptation scheme of F can be formulated as

$$F_i = max(F_1,\ F_2) \tag{13}$$

where $F_1 = F_{max} \cdot (1 - e^{-\Delta f_i})$ and $F_2 = F_{max} \cdot \left(\frac{\Delta f_i}{\lambda + \Delta f_i} \right)$.

The values of CR_i have also been adapted in each generation based on the fitness values obtained for the donor vectors. As evidence of change, variable $\Delta f_{donor\_i} = |f(v_i) - f(x_{best})|$ has been used as a measurement. If the value of Δf_{donor_i} is low, then the value of CR should be higher because the donor vector is close to the best individual located in the current population, and thus it possesses good genetic components to be propagated to the offspring. On the contrary, when the value of Δf_{donor_i} is high, then the value of CR should be lower. Another case is when the donor vector has better fitness value than x_{best} in the same population. In this case, the CR_i should have been assigned to a very high value. Thus, the scheme for determining CR_i can be formulated as

$$\begin{aligned} &if \quad f(v_i) \le f(x_{best}) \\ &CR_i = CR_{const} \\ &else \\ &CR_i = CR_{min} + \frac{(CR_{max} - CR_{min})}{1 + \Delta f_{donor\_i}} \end{aligned} \tag{14}$$

where CR_{const} has been assigned to a high constant value of 0.95. Eq. (14) clearly shows that when the value of $\Delta f_{donor\_i} \to 0$, then CR_i moves towards CR_{max} which is a high value and when $\Delta f_{donor\_i} \to \infty$, then CR_i moves towards CR_{min} which is a lower value.

5.1.1.4. GADE. GADE algorithm proposes an effective greedy mechanism to dynamically update the values of F and CR during the evolution. The main idea of this adaptive strategy is that at every learning period (LP), the current F and CR values are compared with their neighboring setting values and then the settings with the best solution are selected. GADE algorithm has a very simple adaptation procedure that can be easily integrated with any DE strategy such as DE/rand/1. It starts with initializing the values of F and CR_m to 0.5 for both parameters. In each generation, the values of F_i are selected randomly from the set $Z_F = \{F - d_1,\ F,\ F + d_1\}$, and the values of CR_i are selected from the equation below.

$$CR_i = randc_i(\mu_{CR},\ 0.2) \tag{15}$$

where the value of the mean distribution, μ_{CR} is selected randomly from the set $Z_{CR} = \{CR_m - d_2,\ CR_m, CR_m + d_2\}$. The two sets, Z_F and Z_{CR} represent the neighboring setting values. After a specified learning period ($G\%LP$) the values of F and CR_m are then updated as follows:

$$F = \underset{x \in Z_F}{\text{argmax}} PR(x) \tag{16}$$

$$CR_m = \underset{y \in Z_{CR}}{\text{argmax}} PR(y) \tag{17}$$

where $PR(x)$ and $PR(y)$ are the progress rates that are used as criteria to evaluate and compare the algorithm parameters assignments. PR is the average of the relative improvements (RI) from N tests, using the C assignment to produce trial solutions, and it is calculated as follows:

$$PR(C) = \frac{1}{N} \sum_{j=1}^{N} RI(C,j) \tag{18}$$

$$RI(C,j) = \begin{cases} f(x^k) \cdot 10^n - f(v^k) \cdot 10^n, & if\ f(x^k) \ge f(v^k) \\ 0, & otherwise \end{cases} \tag{19}$$

where $RI(C,j)$ is the relative improvement of the candidate assignment C of either the scaling factor or crossover rate, k is the test index, and n is an integer number used to set the values of $f(x^k) \cdot 10^n$ and $f(v^k) \cdot 10^n$ within $[1,\ 10]$ or $[-10,\ -1]$ ranges.

5.1.2. Adaptive DE with single advanced strategy

5.1.2.1. DESAP algorithm

5.1.2.1.1. Advanced DESAP mutation and crossover schemes. In DESAP the base strategy used is a bit different from the standard DE/rand/1/bin and of some sort similar to the strategy introduced in Ref. [1].

Crossover Scheme: The crossover operator is performed first with some probability, $rand(0,1) < \delta_{r1}$ or $i = j$, where j is a randomly selected variable within individual i. The updating strategy is as follows,

$$x_{child} = x_{r1} + F \cdot (x_{r2} - x_{r3}) \tag{20}$$

The ordinary amplification factor F is set to1, thereby at least one variable in x must be changed. Otherwise the value of x_{child} and its control parameters will be set to the same values associated with x_{r1}.

Mutation Scheme: The mutation stage is implemented with some mutation probability, $rand(0,1) < \eta_{r1}$; otherwise, all the values will remain fixed.

$$x_{child} = x_{child} + randn(0, \eta_{r1}) \tag{21}$$

As can be seen from the equation above, DESAP mutation is not derived from any standard DE strategies.

5.1.2.1.2. DESAP parameter control schemes. DESAP not only updates the values of the mutation and crossover control parameters, η and δ, but, rather, it adjusts the population size Np as well in a self-adaptive manner. All parameters undergo the evolution (i.e. crossover and mutation) in a way analogue to their corresponding individuals. The term δ has the same meaning as CR and η refers to the probability of applying the mutation scheme whereas the ordinary F is kept fixed during the evolution process. Mainly, two versions of DESAP have been applied (Rel and Abs). The population size of both DESAP versions are initialized by generating, randomly, a population of size $(10 \times n)$ initial vectors; where n denotes the number of design variables. The mutation probability η_i and crossover rate δ_i are both initialized to random values generated uniformly between $[0,1]$. The population size parameter Np_i is initialized in DESAP-Abs version to,

$$Np_i = round(initial\ population\ size + randn(0,1)) \tag{22}$$

whereas in DESAP-Rel to,

$$Np_i = rand(-0.5, 0.5) \tag{23}$$

The parameters δ, η and π are updated at the same level with their corresponding individuals using the same crossover and mutation schemes (see Eq. (20) and (21)).

Updating the crossover rate δ

$$\delta_{child} = \delta_{r1} + F \cdot (\delta_{r2} - \delta_{r3}) \tag{24}$$

$$\delta_{child} = randn(0,1) \tag{25}$$

Updating the mutation probability η

$$\eta_{child} = \eta_{r1} + F \cdot (\eta_{r2} - \eta_{r3}) \tag{26}$$

$$\eta_{child} = randn(0,1) \tag{27}$$

Updating the population size Np

$$\text{DESAP} - \text{Abs:} \quad Np_{child} = Np_{r1} + int(F \cdot (Np_{r2} - Np_{r3})) \tag{28}$$

$$\text{DESAP} - \text{Rel:} \quad Np_{child} = Np_{r1} + int(F \cdot (Np_{r2} - Np_{r3})) \tag{29}$$

$$\text{DESAP} - \text{Abs:} \quad Np_{child} = Np_{child} + int(randn(0.5,1)) \tag{30}$$

$$\text{DESAP} - \text{Rel:} \quad Np_{child} = Np_{child} + randn(0,\eta_{r1}) \tag{31}$$

The ordinary amplification factor F is set to 1. The evolution process of DESAP continues until it achieves a pre-specified population size M. Then, the new population size is calculated for the next generation as,

$$\text{DESAP} - \text{Abs:} \quad M_{new} = round\left(\sum_{1}^{M} Np/M\right) \tag{32}$$

$$\text{DESAP} - \text{Rel:} \quad M_{new} = round(M + (Np \times M)) \tag{33}$$

5.1.2.2. JADE algorithm

5.1.2.2.1. Advanced JADE mutation schemes. Different mutation versions of JADE have been proposed in Refs. [148] and [149]. The first new mutation scheme is called DE/current-to-*p*best/1/bin (see Eq. (34)), which has less greedy property than its previous specification scheme DE/current-to-best/1/bin, because it utilizes not only the information of the best individual but also the information of the *p*% good solutions in the current population.

$$v_i^G = x_i^G + F_i \cdot \left(x_{best}^{p,G} - x_i^G\right) + F_i \cdot \left(x_{r1}^G - x_{r2}^G\right), \tag{34}$$

where $p \in (0, 1]$ and $x_{best}^{p,G}$ is a random uniform vector chosen as one of the superior 100*p*% vectors in the current population. The second mutation scheme with an *external archive* is denoted as A, which is an archive introduced to store the recent explored *inferior individuals* that have been excluded from the search process. A is first initialized to be empty. Thereafter, failed solutions in the selection operation of each generation are added to this archive. The new mutation operation is then reformulated as follows:

$$v_i^G = x_i^G + F_i \cdot \left(x_{best}^{p,G} - x_i^G\right) + F_i \cdot \left(x_{r1}^G - \tilde{x}_{r2}^G\right), \tag{35}$$

where x_i^G and x_{r1}^G are generated from P in the same way as in the original JADE, whereas $\tilde{x}_{r2}^G$ is randomly generated from the union, $A \cup P$. Eventually, randomly selected solutions are going to be removed from the archive if its size exceeds a certain threshold to keep the archive within a specified dimension. Another JADE variant has been proposed to further increase the population diversity, and it is named archive-assisted DE/rand-to-*p*best/1 as follows:

$$v_i^G = x_{r1}^G + F_i \cdot \left(x_{best}^{p,G} - x_{r1}^G\right) + F_i \cdot \left(x_{r2}^G - \tilde{x}_{r3}^G\right) \tag{36}$$

5.1.2.2.2. JADE parameter control schemes. JADE updates four control parameters (F, CR, μ_F and μ_{CR}) during evolution.

Mutation factor (F) *and location parameter of mutation probability distribution* (μ_F): The mutation probability F_i is independently generated in each generation for each individual i according to the following formula:

$$F_i = randc_i(\mu_F, 0.1) \tag{37}$$

where $randc_i$ is a Cauchy distribution with location parameter μ_F and scale parameter 0.1. If $F_i \geq 1$, then the value is truncated to be 1 or regenerated if $F_i \leq 0$. The location parameter μ_F is first initiated to be 0.5. In this step, JADE shows some similarities in updating the mean of the distribution, μ_{CR} to the learning style used in Population Based Incremental Learning (PBIL) algorithm [13,14]. The standard version of the PBIL uses learning rate $LR \in (0, 1]$ that must be fixed a priori. By utilizing Hebbian-inspired rule, the difference rate $(1 - LR)$ is then multiplied by the probability vector (PV) that represents the combined experience of the PBIL throughout the evolution, whereas LR is multiplied by each bit (i.e. gene's value) of the current individual(s) used in the updating process. Likewise, JADE updates the mutation distribution mean location, μ_F after accumulating the set of all the successful mutation probabilities F_i's at generation G, denoted by S_F. The new μ_F is updated as

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F), \tag{38}$$

where $mean_L(\cdot)$ is Lehmer mean,

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{39}$$

Crossover probability (CR) *and mean of crossover probability distribution* (μ_{CR}): The crossover probability CR_i is updated independently for each individual according to a normal distribution as,

$$CR_i = randn_i(\mu_{CR}, 0.1), \tag{40}$$

with mean μ_{CR} and standard deviation 0.1 and truncated to the interval $(0, 1]$. The mean μ_{CR} is first initiated to 0.5. Then, μ_{CR} is updated in each generation after accumulating the set of all the successful crossover probabilities CR_i's at generation G, denoted as S_{CR}, hence calculate its $mean_A(S_{CR})$. The new μ_{CR} is updated as

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR}), \tag{41}$$

where c is a positive constant $\in (0, 1]$ and $mean_A(\cdot)$ is the usual arithmetic mean.

5.1.2.3. DEGL (DE with Global and Local Neighborhoods) algorithm

5.1.2.3.1. Advanced DEGL mutation strategy. In this algorithm, a novel mutation strategy named Local and Global Neighborhood-Based Mutations is proposed. This new scheme has been inspired by the fact that a proper balance must be achieved in the search evolution between the two objectives, exploitation and exploration capabilities of the search technique. Two models have been used to achieve the aforementioned objectives: *local neighborhood model* and *global mutation model*. The local model is performed to generate a local donor vector l_i^G of each individual in the current population as

$$l_i^G = x_i^G + \alpha \cdot \left(x_{n\_best_i}^G - x_i^G\right) + \beta \cdot \left(x_p^G - x_q^G\right) \tag{42}$$

where $x_{n\_best_i}^G$ is the best individual in the neighborhood of x_i^G, x_p^G and x_q^G; $p, q \in [i - k, i + k]; p \neq q \neq i; k$ is the neighborhood radius with value from 0 to $(Np - 1)/2$. This strategy satisfies the first objective (i.e.

improve the local search). A global donor vector g_i^G is also generated for each individual as

$$g_i^G = x_i^G + \alpha \cdot \left(x_{g\_best}^G - x_i^G\right) + \beta \cdot \left(x_{r1}^G - x_{r2}^G\right) \tag{43}$$

where $x_{g\_best}^G$ is the best individual in the entire current population; $r1$, $r2 \in [1, 2, \ldots, Np]$, $r1 \neq r2 \neq i$. This strategy satisfies the second objective (i.e. improve the global search). In both models, α and β are the scaling factors. These two models are then combined to formulate a general model that produces the actual donor individual as

$$v_i^G = w \cdot g_i^G + (1 - w) \cdot I_i^G \tag{44}$$

where w is a scalar factor with a value in $(0, 1)$.

5.1.2.3.2. DEGL parameter control schemes. DEGL disposes four new control parameters α, β, w, and k. However, the most effective parameter control among them is w because it creates the balance between the exploitation and exploration capabilities in DEGL. The number of control parameters has been subsequently reduced because $\alpha = \beta = F$ has been considered. In DEGL, three adaptive schemes have been proposed to update the value of w.

1- Increasing Weight Factor: This scheme increases the value of w from 0 to 1 whenever the search process requires. When the search moves towards exploration the value of w is decreased to 0. When the exploitation is required, the value of w is increased towards 1. For this reason, the initial vectors in the first generation are assigned to $w = 0$, to activate the exploration moves; the contrary is applied at the final stages of the evolution in which exploitation is required. The middle value of w, 0.5 results in balanced performance in DEGL. Two schemes have been proposed for this purpose.

Linear increment: w is increased linearly from 0 to 1.

$$w^G = \frac{G}{G_{max}} \tag{45}$$

Exponential increment:w is increased exponentially from 0 to 1.

$$w^G = \exp\left(\frac{G}{G_{max}} \cdot ln(2)\right) - 1 \tag{46}$$

2- Random Weight Factor: For each individual the value of w is generated using uniformal random number, $w_i^G \sim (0, 1)$.

3- Self-Adaptive Weight Factor: As previously discussed in Section 3, when the value of the control parameter w_i is embedded in the individual representation and undergoes evolution, this is known as self-adaptive setting, which is expressed as follows:

$$w_i^{'G} = w_i^G + F \cdot \left(w_{g\_best}^G - w_i^G\right) + F \cdot \left(w_{r1}^G - w_{r2}^G\right) \tag{47}$$

where $w_{g\_best}^G$ is the weight factor associated with the best individual in the current population. w_i^G are randomly initialized in the first generation to values in the range $(0.0, 1.0)$ and then updated during generations.

5.1.2.4. MDE_pBX algorithm

5.1.2.4.1. Advanced MDE_pBX mutation and crossover schemes. Mutation Scheme: The new proposed mutation scheme DE/current-to-gr_{best}/1/bin, utilizes the best individual $x_{gr_{best}}^G$ chosen from the q% group of individuals that were *randomly* selected from the current population for each target vector. The group size q of the MDE_pBX varies from 5% to 65% of the Np. The new scheme can be described as

$$v_i^G = x_i^G + F_i \cdot \left(x_{gr_{best}}^G - x_i^G + x_{r1}^G - x_{r2}^G\right), \tag{48}$$

where x_{r1}^G *and* x_{r2}^G are two different individuals randomly selected from the current population, and they are mutually different from the running individual x_i^G and $x_{gr_{best}}^G$.

Crossover Scheme: The new proposed recombination scheme *p-Best* has been defined as a greedy strategy, which is based on the incorporation of a randomly selected mutant vector perturbed by one of the p top-ranked individual selected from the current population to yield the trial vector at the same index. Throughout evolution, the value of parameter p is reduced linearly in an adaptive manner (see Eq. (55)).

5.1.2.4.2. MDE_pBX parameters control schemes. Modifications applied to the adaptive schemes in MDE_pBX: The scalar factor F_i and crossover rate CR_i of each individual are both altered independently in each generation using JADE schemes (see Eqs. (37) and (40)). In MDE_pBX, both μ_F and μ_{CR} are subscribed to the same rule of adjusting. Firstly, their values are initialized to 0.5 and 0.6 respectively. Subsequently, they are updated in each generation as follows:

$$\mu_F = w_F \cdot \mu_F + (1 - w_F) \cdot mean_{pow}(F_{success}) \tag{49}$$

$$\mu_{CR} = w_{CR} \cdot \mu_{CR} + (1 - w_{CR}) \cdot mean_{pow}(CR_{success}) \tag{50}$$

where a set of successful scalar factors $F_{success}$ and a set of successful crossover probability $CR_{success}$ are generated from the current population. $|.|$ stands for the cardinality of each successful set. Variable n is set to 1.5 then the mean power $mean_{pow}$ of each set is calculated as follows:

$$mean_{Pow}(F_{success}) = \sum_{x \in F_{success}} \left(x^n / |F_{success}|\right)^{\frac{1}{n}} \tag{51}$$

$$mean_{Pow}(CR_{success}) = \sum_{x \in CR_{success}} \left(x^n / |CR_{success}|\right)^{\frac{1}{n}} \tag{52}$$

including the calculation of the weight factors w_F and w_{CR},

$$w_F = 0.8 + 0.2 \times rand(0, 1) \tag{53}$$

$$w_{CR} = 0.9 + 0.1 \times rand(0, 1) \tag{54}$$

the μ_F and μ_{CR} are formulized. As seen from Eqs. (53) and (54), the value of w_F uniformly and randomly varies within the range $[0.8, 1]$, whereas the value of w_{CR} uniformly and randomly varies within the range $[0.9, 1]$. The small random values used to perturb the parameters μ_F and $mean_{Pow}$ reveal an improvement in the performance of MDE_pBX as it emphasizes slight variations on these two parameters each time F is generated.

Crossover amplification factor (p): Throughout evolution, the value of parameter p is reduced linearly in the following manner:

$$p = ceil\left[\frac{Np}{2} \cdot \left(1 - \frac{G - 1}{G_{max}}\right)\right] \tag{55}$$

where $ceil(y)$ is the "*ceiling*" function that outputs the smallest integer $\geq y$. The reduction monotony of the parameter p creates the required balance between the exploration and exploitation capabilities of DE.

5.1.2.5. p-ADE algorithm

5.1.2.5.1. Advanced p-ADE mutation scheme. A new mutation strategy called DE/rand-to-best/pbest/bin is used; which is, essentially, based on utilizing the best global solution and the best previous solution of each individual that are involved in the differential process, thus bringing in more effective guidance information to generate new individuals for the next generation. The detailed operation is as follows:

$$v_i^G = W_i^G \cdot x_{r1}^G + K_i^G \cdot \left(x_{best}^G - x_i^G\right) + F_i^G \cdot \left(x_{pbesti}^G - x_i^G\right) \tag{56}$$

where x_{best}^G denotes the best individual in the current generation G. x_{r1}^G is a random generated individual where $r1 \in [1, 2, \ldots, Np]$ and $r1 \neq i$. x_{pbesti}^G denotes the best i^{th} previous individual picked up from the previous generation. The mutation's control parameters W_i^G, K_i^G, and F_i^G of the i^{th}

individual are updated using a dynamic adaptive manner. The most remarkable merit of this mutation technique is the inclusion of three different working parts at the same time.

Inertial Part (*Inheriting part*) represented by $(W_i^G \cdot x_{r1}^G)$ where the current individual, v_i^G inherits traits from another individual at generation G.

Social Part (Learning Part) represented by $K_i^G \cdot (x_{best}^G - x_i^G)$ where the current individual, v_i^G gains information from the superior individual in the current generation. G.

Cognitive Part (Private Thinking) represented by $F_i^G \cdot (x_{pbesti}^G - x_i^G)$ where the current individual, v_i^G reinforces its own perception through the evolution process.

The high values of both the inertial and the cognitive part play a key role in intensifying the exploration searching space, thus improving its ability for finding the global solution. While the large values of the social part promotes connections among individuals, thus resulting to speed up the convergence rate. In *p*-ADE there is an additional mechanism which is called *classification mechanism*. This *classification mechanism* is coupled with the mutation scheme to be implemented on the whole population at each generation. Accordingly, the new mechanism divides the population's individuals into three classes:

Superior individuals: The first individuals' category where the fitness values of these individuals fall in the range $f_i - f_{mean} < -E(f^2)$, where f_{mean} is the mean fitness values and $E(f^2)$ is the second moment of the fitness values of all individuals in the current generation. In this case, the exploration ability of the search process is enhanced by further intensifying the inertial and cognitive parts in order to increase the likelihood of the excellent individual to find the global solution in its neighborhood area. The corresponding individual is generated as follows:

$$v_i^G = W_i^G \cdot x_{r1}^G + F_i^G \cdot \left(x_{pbesti}^G - x_i^G\right) \tag{57}$$

Inferior individuals: The second individuals' category where the fitness values of these individuals fall in the range $f_i - f_{mean} > E(f^2)$. The individual in this case has poor traits since its place in the search space is far away from the global optimum. Therefore, the exploration search ability is also intensified for rapid convergence rate. The corresponding individual is generated as follows:

$$v_i^G = W_i^G \cdot x_{r1}^G + K_i^G \cdot \left(x_{best}^G - x_i^G\right) \tag{58}$$

Middle Individuals: The third individuals' category where the fitness values of these individuals fall in the range $-E(f^2) < f_i - f_{mean} < E(f^2)$. The individuals in this category are not superior nor are they inferior; therefore, the complete perturbation scheme (see Eq. (56)) should be implemented entirely for further improving both the exploitation and exploration abilities.

5.1.2.5.2. p-ADE parameter control schemes. *p*-ADE comprises four control parameters involved in the search process, including three mutation scheme parameters (W, F and K) and crossover rate CR. A dynamic adaptive scheme has been proposed to jointly update the four parameters through the run as follows:

$$W_i^G = W_{min} + (W_{max} - W_{min}) \times \left(\left(2 - exp\left(\frac{G}{G_{max}} \times ln(2)\right)\right) \times \frac{1}{2} + \frac{f_i^G - f_{min}^G}{f_{max}^G - f_{min}^G} \times \frac{1}{2}\right) \tag{59}$$

$$K_i^G = K_{min} + (K_{max} - K_{min}) \times \left(\left(exp\left(\frac{G}{G_{max}} \times ln(2)\right) - 1\right) \times \frac{1}{2} + \frac{f_i^G - f_{min}^G}{f_{max}^G - f_{min}^G} \times \frac{1}{2}\right) \tag{60}$$

$$F_i^G = F_{min} + (F_{max} - F_{min}) \times \left(\left(2 - exp\left(\frac{G}{G_{max}} \times ln(2)\right)\right) \times \frac{1}{2} + \frac{f_i^G - f_{min}^G}{f_{max}^G - f_{min}^G} \times \frac{1}{2}\right) \tag{61}$$

$$CR_i^G = CR_{min} + (CR_{max} - CR_{min}) \times \left(\left(2 - exp\left(\frac{G}{G_{max}} \times ln(2)\right)\right) \times \frac{1}{2} + \frac{f_i^G - f_{min}^G}{f_{max}^G - f_{min}^G} \times \frac{1}{2}\right) \tag{62}$$

As seen from the above equations, the main adaptive scheme is equally captive to the influence of the number of generations achieved, as well as the fitness values. Technically, the value of each control parameter varies within its specified range, $W \in [0.1,\ 0.9]$, $K \in [0.3,\ 0.9]$, $F \in [0.3,\ 0.9]$ and $CR \in [0.1,\ 0.9]$ during the run.

5.1.2.6. SHADE algorithm. SHADE is an improved version of JADE algorithm specifically in updating the control parameter values of F and CR. SHADE kept the mutation scheme of JADE w archive (see Eq. (35)) as it is but modified the adaptation strategy of the F and CR mean values (μ_F and μ_{CR}). The main idea of SHADE is to no longer generate the values of F_i and CR_i from μ_F and μ_{CR} as two distribution mean values are updated once in each generation but are generated from a historical memory (M_F, M_{CR}) that stores the mean values of S_F and S_{CR} each time these two sets are updated with new values of μ_F and μ_{CR}, respectively. This new modification improves the performance of DE in general and JADE in specific due to the inclusion of more values in $M_{F,k}$ and $M_{CR,k}$ which leads the search towards different search regions especially when dealing with problems of different scenarios.

5.1.2.6.1. SHADE Parameter Control Schemes. SHADE maintains a historical memory with H entries for the mean values of S_F and S_{CR} and denoted as $M_{F,k}$ and $M_{CR,k}$, respectively, and $k \in [1, H]$. In the first generation, the values of all $M_{F,k}$ and $M_{CR,k}$ are initialized to 0.5. When generating the values of F_i and CR_i in each generation, a new adaptation scheme is proposed as follows:

$$F_i = randc_i(M_{F,r_i}, 0.1) \tag{63}$$

$$CR_i = randn_i(M_{CR,r_i}, 0.1) \tag{64}$$

where r_i is randomly selected index within $[1, H]$. In case the values of F_i and CR_i exceed to 0 or 1, the same adjusting procedure in JADE is followed. Before the end of each generation, the successful values of F_i and CR_i are stored in S_F and S_{CR}, respectively. The values of these two sets are then used to update the historical memory of M_F and M_{CR} as follows:

$$M_{F,k}^{G+1} = \begin{cases} meanw_L(S_F) & if\ \ S_F \neq \varnothing \\ M_{F,k}^G & otherwise \end{cases} \tag{65}$$

$$M_{CR,k}^{G+1} = \begin{cases} meanw_A(S_{CR}) & if\ \ S_{CR} \neq \varnothing \\ M_{CR,k}^G & otherwise \end{cases} \tag{66}$$

The value of k is first initialized to 1 and then incremented by 1 each time an element is added to the memory. If k reaches a threshold (it is H in SHADE), then k is set to 1. The weighted Lehmer mean $meanw_L(S_F)$ is computed using the equation below.

$$meanw_L(S_F) = \frac{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}^2}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}} \tag{67}$$

To avoid the bias towards small values that may be generated because of the arithmetic mean used in Eq. (41), Peng et al. [97] suggested using the following weighted mean [97]:

$$meanw_A(S_{CR}) = \sum_{k=1}^{|S_{CR}|} w_k \cdot S_{CR,k} \tag{68}$$

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k} \tag{69}$$

where $\Delta f_k = |f(u_k^G) - f(x_k^G)|$. Finally, in JADE the value of p (see Eq. (34)) is kept fixed during the run. In the SHADE algorithm, the value of this parameter is associated with each x_i and altered during generations as follows:

$$p_i = rand[p_{min}, 0.2] \tag{70}$$

where $p_{min} = 2/Np$.

5.1.2.7. L-SHADE Algorithm. This algorithm is an improved version of SHADE algorithm in which a deterministic population resizing scheme is incorporated to dynamically reduce the size of population according to a linear function. This scheme is named as linear population size reduction (LPSR), and it is a specialized version of simple variable population sizing (SVPS) scheme proposed in Ref. [69]. The main concept of LPSR is to continually reduce the population size Np linearly from the initial size which is Np_{init} to the target population size which is Np_{min} using the function below.

$$Np^{G+1} = round\left[\left(\frac{Np_{min} - Np_{init}}{MAX\_NFE}\right) \cdot NEF + Np_{init}\right] \tag{71}$$

Where MAX_NFE is the maximum number of fitness evaluation, and NFE is the current number of fitness evaluation. Based on the equation above, the worst ranking vectors $(Np^G - Np^{G+1})$ are deleted whenever $Np^{G+1} < Np^G$.

L-SHADE has also suggested a slight improvement to the scheme of generating the values of CR_i, in which a terminal value "$\perp$" is assigned to the M_{CR,r_i} when $max(S_{CR}) = 0$ (i.e. all elements in S_{CR} are 0). This modification has led to the following updated scheme:

$$CR_i = \begin{cases} 0 & if\ M_{CR,r_i=\perp} \\ randn_i(M_{CR,\,r_i},\ 0.1) & otherwise \end{cases} \tag{72}$$

The 0 value assigned to CR_i enforces only one parameter change in the original crossover scheme of DE, which implies to slow down the convergence of the algorithm, and it is effective in the multimodal problems. Finally, the weighted Lehmer mean is used to calculate the mean values for both S_F and S_{CR} (i.e. S) sets using the equation below for both M_F and M_{CR} .

$$meanw_L(S) = \frac{\sum_{k=1}^{|S|} w_k \cdot S_k^2}{\sum_{k=1}^{|S|} w_k \cdot S_k} \tag{73}$$

$$w_k = \frac{\Delta f_k}{\sum_{j=1}^{|S|} \Delta f_j} \tag{74}$$

$$\Delta f_k = \left|f\left(u_k^G\right) - f\left(x_k^G\right)\right| \tag{75}$$

5.1.2.8. EsDEr-NR Algorithm. In EsDE$_r$-NR algorithm, the advanced mutation strategy proposed in JADE is adopted (see Eq. (35)). EsDE$_r$-NR has three improved parts: ensemble sinusoidal parameter adaptation scheme, niching-based population size reduction, and restart method. They are presented briefly below.

5.1.2.8.1. EsDE$_r$-NR Parameter Control Schemes. As previously suggested in Ref. [11], in every generation in the first half of the evolution process $G \in \left[1,\ \frac{G_{max}}{2}\right]$ the values of F_i are updated using either one of the two sinusoidal adaptation approaches as presented below. The preference between these two approaches is implemented randomly.

sinusoidal decreasing adjustment

$$F_i^G = \frac{1}{2} \cdot \left(sin(2\pi \cdot freq \cdot G + \theta) \cdot \frac{G_{max} - G}{G_{max}} + 1\right) \tag{76}$$

sinusoidal increasing adjustment

$$F_i^G = \frac{1}{2} \cdot \left(sin\left(2\pi \cdot freq_i^G \cdot G + \theta\right) \cdot \frac{G}{G_{max}} + 1\right) \tag{77}$$

where θ is the phase shift value, $freq$ is set to a fixed value, and $freq_i^G$ is set automatically using an adaptive scheme as expressed below.

$$freq_i^G = randc_i(\mu_{freq},\ 0.1) \tag{78}$$

Where μ_{freq} is the Cauchy distribution's mean with 0.1 as a standard deviation. The value of μ_{freq} is calculated using the usual arithmetic mean of the values in S_{freq} set of the successful mean frequinces.

For the second half of the evolution, where $G \in \left[\frac{G_{max}}{2}, G_{max}\right]$, the values of F_i are adapted in the same way suggested in Ref. [125] using Eq. (63). For the values of CR_i, they are adapted during the run using Eq. (64). Both F and CR values adopted Eqs. (73)–(75) of L-SHADE to calculate the distribution mean values.

5.1.2.8.2. EsDE$_r$-NR Restart Method. A restart mechanism is activated at the late stages of the evolution when the population size is reduced and the diversity of solutions is degraded. This mechanism is activated when the size of the population exactly reaches 20 individuals. Half of these individuals are then re-initialized using Eq. (1) and further modified using a Gaussian walk formula as follows:

$$y_i = Gaussian(\mu,\ \sigma) + (rand(0,1) \cdot x_{best} - rand(0,1) \cdot x_i) \tag{79}$$

where the mean, μ is set to x_{best} which is the best individual in the new generated group of individuals; x_i is the i^{th} individual in the same group of the new individuals; and σ is computed as follows:

$$\sigma = \left|\frac{log(G)}{G} \cdot (x_i - x_{best})\right| \tag{80}$$

After these computations, a replacement occurs between the worst individuals and the new individuals in the G_{LS} generations if the new individuals are better than the old individuals.

5.1.2.8.3. EsDE$_r$-NR Niching-based population size reduction scheme. - After half of the function evaluations have been consumed, a novel population size reduction mechanism is activated to ensure that all individuals will have equal chance to evolve before the reduction is performed. From the start, each individual x_i^G will have its Euclidian distance, ED_i from the best individual, x_{best}^G calculated as follows:

$$\begin{aligned} ED_i &= ED_i\left(x_i^G,\ x_{best}^G\right) \\ &= \sqrt{\left(x_{i,1}^G - x_{best,1}^G\right)^2 + \left(x_{i,2}^G - x_{best,2}^G\right)^2 + \ldots + \left(x_{i,D}^G - x_{best,D}^G\right)^2} \end{aligned} \tag{81}$$

These individuals are then sorted according to their ED values and divided into two niche groups. The first niche contains the best individuals and the individuals in their neighborhood area (i.e. individuals with the smallest ED values). The second individual contains the remaining individuals. These two niches have equal size which is $\frac{Np}{2}$, and since Np is decreased after the first half of the evolution in each generation; the size of the two niches is also linearly decreased using Eq. (71) in L-SHADE. The individuals in the second niche should be eliminated

after satisfying different criteria.

5.2. DE with adaptive parameters and multiple strategies

In this subsection, the main characteristics and mechanisms of 7 adaptive DE versions are stated in detail on the basis of parameter adaptive schemes and multiple adaptive DE strategies.

5.2.1. Adaptive DE with multiple standard strategies

5.2.1.1. SaDE Algorithm

5.2.1.1.1. SaDE Adaptive Strategies. The main feature of SaDE is to automatically adapt multiple standard DE mutation strategies (DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin, and DE/current-to-rand/1 with no crossover) and update the corresponding control parameters during the evolution process using parameter, $p_k (k = 1, 2, 3, 4)$.

Determine the probability of applying each candidate strategy to the current population (p_k): Initially, the probabilities of applying each scheme p_k^G is set to $1/K$ to assign an equal probability for all strategies. The probability of applying each strategy is then updated every 50 generations in the following manner:

$$p_k^G = \frac{S_k^G}{\sum_{k=1}^{K} S_k^G} \tag{82}$$

where

$$S_k^G = \frac{\sum_{g=G-LP}^{G-1} ns_k^g}{\sum_{g=G-LP}^{G-1} ns_k^g + \sum_{g=G-LP}^{G-1} nf_k^g} + \varepsilon, \quad for\ k = 1, 2, \ldots, K;\ G > LP$$

where K is the number of strategies available for perturbation. LP is the period assigned for learning in which the learning process is activated only when $G > LP$; LP has been set to 50 generations. ns_k^G (*Success Memory*) and nf_k^G (*Failure Memory*) are both memories generated by the k^{th} strategy and used to record the number of trial vectors that have succeeded or failed to enter the search process, respectively. The small value, that is, $\varepsilon = 0.001$, is added to avoid the possibility of the null rate of success. Once the sizes of these memories reach a certain threshold, (i.e. after LP iterations), all previous records will be eliminated from these memories, (i.e. ns^{G-LP} and nf^{G-LP}) to allow those vectors that are generated in the current iteration to be stored. Finally, S_k^G is divided by $\sum_{k=1}^{K} S_k^G$ to guarantee that the resultant p_k^G is always summed to 1.

5.2.1.1.2. SaDE Parameters Control Schemes. Set the mutation factor F_i *values* as independently generated in each generation according to Gaussian distribution with mean 0.5 and standard deviation 0.3 as follows:

$$F_i = randn(0.5, 0.3) \tag{83}$$

Accordingly, both the local (with small F_i values) and global (with large F_i values) search abilities will be kept throughout the evolution, hence to generate good mutant vectors.

Crossover Probability (CR_i) *and the Mean Crossover Probability Distribution* (CR_m)*:* The strategy of controlling the crossover probability CR is an adaptive controlled scheme. It starts with independently generating crossover probabilities CR_i under Gaussian distribution with mean CR_m and standard deviation 0.1 as follows:

$$CR_i = randn(CR_m, 0.1) \tag{84}$$

The CR_i values will remain fixed for five generations before the next generation has launched. Throughout these generations, CR_i values that are associated with successful trial vectors are recorded. By contrast, the value of the median CR_m is first initialized to 0.5 and then updated every 25 generations based on the successful CR_i values as follows:

$$CR_m = \frac{1}{K} \sum_{k=1}^{K} CR_{suc}(k), \tag{85}$$

where K denotes the number of successful CR_i values accumulated over 25 generations, and CR_{suc} is the k^{th} CR successful value.

5.2.1.2. EPSDE Algorithm

5.2.1.2.1. EPSDE Parameters Control Schemes and Strategies. EPSDE is unlike other adaptive DE variants, it is an ensemble of mutation strategies and parameter values of DE. EPSDE does not involve a certain equation to modify the values of the control parameters. Rather, it assigns for each member of the initial population a mutation strategy randomly selected from a pool of mutation strategies with diverse characteristics and randomly takes values for the associated parameter from a pool of values. Throughout evolution, the population members that produce individuals better than the target vectors, their mutation strategies and associated parameter values are retained for the next generation, whereas those that fail to produce better individuals are reinitialized with a new mutation strategy and associated parameter values from the respective pools or from the successful combinations stored with equal probability. EPSDE comprises two pools.

Pool of mutation strategies: This pool includes the DE strategies that are involved in the evolution. These strategies have been selected with the following diverse characteristics:

1. Strategies relying on the best individual in the current population, DE/best/2/bin
2. Strategies bearing strong exploration capabilities, DE/rand/1/bin
3. Strategies being rotational invariant without crossover, DE/current-to-rand/1

Pool of parameter control values: In this pool, the two crucial parameter values (F and CR) are set to different ranges. The pool of CR values is taken in the range $0.1 - 0.9$ in steps of 0.1. The pool of F values is taken in the range $0.4 - 0.9$ in steps of 0.1.

5.2.1.3. NRDE Algorithm. In NRDE algorithm different mechanisms have been proposed to adapt the parameter control values of F and CR as well as the two standard mutation strategies of DE (DE/rand/1 and DE/best/1). The work proposed in NRDE is an extended version of a published work [32] and as a new attempt after few attempts on investigating the performance of DE on strongly noisy problems such as the one presented in Ref. [25].

5.2.1.3.1. NRDE Parameters Control Schemes. The control parameters F and CR of NRDE are altered using a simple mechanism that requires no prior knowledge or feedback from the search space during the run such as learning strategy. For each offspring to be generated, the value of F is selected randomly from the set of feasible values $\{0.5,\ 2\}$. At the same time, the value of CR is selected randomly from the interval $[0, 1]$ for each donor vector to be generated.

5.2.1.3.2. NRDE Adaptive Mutation Strategies. NRDE proposes a simple mechanism to switch between two alternative DE mutation strategies (DE/rand/1 and DE/best/1) without using a pool of strategies. The selection between these two strategies is implemented in an equally probabilistic switch which is 0.5. The reason behind the use of these two strategies is DE/rand/1 is known as a very effective strategy in creating population with high diversity (i.e. highly explorative), whereas DE/best/1 focuses on creating solutions by perturbing the best solutions found so far in various search directions (i.e. highly exploitative).

5.2.1.3.3. NRDE Adaptive Crossover Strategies. Using the same switching manner in mutation strategies, NRDE proposes a simple mechanism to switch between two alternative DE crossover strategies in an equally probabilistic manner which is 0.5. One of these strategies is newly proposed in this algorithm as follows:

$$u_{j,i}^{G} = \begin{cases} b \cdot x_{j,i}^{G} + (1-b) \cdot v_{j,i}^{G} & if \;\; rand_{j,i} \leq 0.5 \;\; or \;\; j = j_r \\ x_{j,i}^{G} & otherwise \end{cases} \tag{86}$$

where the value of parameter b is chosen randomly from three values 0.2, 0.5, and 0.9. The main reason behind the use of these three values is because when b is small (i.e. 0.2), a large portion relative to the donor vector is used. When b is in the middle (i.e. 0.5), an equal portion is taken from both the target and donor vector. Finally, when the value of b is 0.9, a gene component is given very near to the target vector. The second crossover strategy is the standard DE crossover defined in Eq. (7).

5.2.1.3.4. NRDE and Self-Adaptive Individual Selection Strategy. NRDE adopted three cases in the survival selection step instead of two as in most of DE variants because the main objective of NRDE is to tackle noise or uncertainties in the optimization problems, and the standard greedy selection strategy may fail to distinguish whether the selected solution is absolutely superior or inferior in comparison to its parent. These cases are defined as follows:

$$x_i^{G+1} = \begin{cases} u_i^G, & if \;\; \dfrac{f(u_i^G)}{f(x_i^G)} \leq 1, \\ u_i^G, & if \;\; \dfrac{f(u_i^G)}{f(x_i^G)} \leq 1+\delta, \\ x_i^G, & otherwise \end{cases} \tag{87}$$

where δ is adapted in a self-adaptive manner using the following formula:

$$\delta^{G+1} = \delta^{G} \cdot (1-\alpha)^{i} \tag{88}$$

where i is the generation index, and the value of α is very small which is 0.04. δ is initially set to 0.3.

5.2.2. Adaptive DE with multiple advanced strategies

5.2.2.1. SaDE-MMTS Algorithm. SaDE-MMTS has been proposed to enhance the performance of the standard SaDE algorithm; by *incorporating* SaDE with the JADE mutation strategy and *integrating* it with the modified multi-trajectory algorithm (MMTS) to solve problems with complex characteristics and high dimensionality. This integration can be encapsulated into three main parts: *SaDE-MMTS = JADE mutation scheme + SaDE algorithm + MMTS method* (Local Search), as follows:

5.2.2.1.1. SaDE-MMTS Advanced Adaptive DE Strategies. JADE mutation strategy with external archive (see Eq. (35)) is adopted and engaged with three crossover operators (binomial and exponential), and no crossover option as well, to generate the trail vectors for the new population. The expected number of perturbation strategies is three, and they are applied according to the strategy probability as in the SaDE algorithm. The selection of the mutation strategy is according to the probability p_k^G of applying each JADE with archive strategy in the current population (see Eq. (82)).

5.2.2.1.2. SaDE Parameters Control Schemes. The control parameters F and CR are updated through evolution in the same manner used in SaDE (see Eqs. (83) and (84)).

5.2.2.1.3. MMTS method. The original multi-trajectory (MTS) algorithm [127,128] is first proposed to solve large scale global optimization problems. The main underlying idea of this algorithm is the employment of randomly selected search combinations (i.e. agents) uniformly distributed over the whole search space to seek out for better solutions. Each combination applies one of the three candidate local search methods that better fits the search space characteristics of a solution's neighborhoods. These combinations are generated using the basic orthogonal array $OA_{n\times k}$, where n is the number of testing experiments and k is the number of factors in each experiment.

5.2.2.2. SaM. SaM is a strategy adaptation mechanism that can be integrated with any DE variant to make it strategy adaptive. SaM creates a pool of strategies and selects the candidate strategy to be applied on the running individual x_i from this pool according to Eq. (89).

$$S_i = \eta_i \times K + 1 \tag{89}$$

where $\eta_i \in [0,1)$ is a strategy parameter control variable. K is the total number of strategies in the pool and $S_i = \{1,2,\ldots,K\}$ the selected DE strategy. For example, suppose $K = 4$ and at a certain generation $\eta_i \in [0, 0.25)$, then based on the calculation of Eq. (89) the value of S_i is 1.

SaM has suggested three approaches to update the value of η_i during evolution. In this study, the first approach has been considered which is inspired by the parameter adaptation equation of JADE. For each individual x_i at generation G, a new value for η_i is generated as follows:

$$\eta_i = \begin{cases} randn_i & G = 1 \\ randn_i(\mu_s,\ 0.1) & otherwise \end{cases} \tag{90}$$

where $randn_i(\mu_s,\ 0.1)$ indicates a normal random distribution of mean μ_s and standard deviation 0.1. The mean μ_s is initialized to be 0.5 and then updated at the end of each generation in the same adaptation equation used in JADE (see Eq. (41)) as follows:

$$\mu_s = (1-c) \times \mu_s + c \times mean_A(H_s) \tag{91}$$

where H_s denotes as the set of all successful DE strategy parameters η_i's at generation G.

SaM mechanism has been applied on JADE strategies to create a new approach called SaJADE. SaJADE employed a pool of different JADE strategies: (1) DE/current-to-*p*Best/1/bin with no archive; (2) DE/current-to-*p*Best/1/bin with archive; (3) DE/rand-to-*p*Best/1/bin with no archive; (4) DE/rand-to-*p*Best/1/bin with archive. The parameter adaptive equations used to update the values of F_i and CR_i are also JADE schemes as in Eq. (37)–(41). For updating the value of CR_i, all strategies use Eq. (40). The difference is in updating the value of F_i in which strategies 1 and 3 use Eq. (37); whereas strategies 2 and 4 use a modified version of Eq. (37) which is normal distribution generator as follows:

$$F_i = randn_i(\mu_F, 0.1) \tag{92}$$

5.2.2.3. EADE Algorithm

5.2.2.3.1. Advanced Adaptive EADE Mutation scheme. This algorithm proposes a novel mutation strategy that contributes to balance the tendency of both the exploration and exploitation capabilities of DE, as well as enhancing its convergence rate. This new strategy has two aims to guide the search process towards fine evolution. The first aim is to preserve the exploration capability during evolution and avoid premature convergence at local optima by preventing the individuals from not always following the best individual in the current population. The second is to preserve at the same time the exploitation capability by preventing the search process from following the bottom worst individual, thus, guiding the search towards promising sub-regions in the search space. These two objectives can be met by applying the new mutation strategy as follows:

$$v_i^{G+1} = x_r^G + F_1 \cdot \left(x_{p\_best}^G - x_r^G\right) + F_2 \cdot \left(x_r^G - x_{p\_worst}^G\right) \tag{93}$$

where x_r^G is randomly chosen individual from the middle ($Np -$

$2(100p\%)$) of the current population; $x^G_{p\_best}$ is a randomly chosen individual from the top $100p\%$ individuals; $x^G_{p\_worst}$ is a randomly chosen individual from the worst $100p\%$ individuals. In both cases, $p \in (0, 1]$; and F_1 and F_2 are uniformly and independently generated in the range $(0, 1)$. From the equation above, sharing information from the best to worst individuals in the same population helps direct the search to different sub-regions in the search space by balancing the local exploitation and global exploration tendencies in the DE population. In EADE, this new mutation strategy is applied with probability of 0.5 in which the standard DE/rand/1/bin is applied otherwise.

5.2.2.3.2. EADE Parameter Control Schemes. EADE proposes a novel adaptive scheme for gradual change of the values of CR_i that can take benefit from the experience of the individuals in the search space during evolution which, in turn, can considerably balance the common trade-off between the population diversity and convergence speed. This scheme can be implemented by applying a simple and straightforward procedure. This procedure has specific parameters to be defined and set to values before and during the run. The first parameter is the pool $A = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.95]$ of pre-specified values for CR. These values are selected randomly from A for each CR_i, using uniform distribution during evolution. A is updated during and after a pre-defined learning period ($LP = 10\%\ G_{max}$). Another parameter is $CR\_Flag\_List_i$, which is a flag that is set to either 0 or 1 to indicate whether the target vector is better than the trial vector or the opposite, respectively. $Failure\_counter\_list_i$ is a parameter used to monitor the progress of the individuals' fitness values after LP completion. If no improvement is observed in the fitness, then the counter of that individual is increased by 1. This process is repeated until $Max\_failure\_counter = 20$ is reached. Another important parameter is $CR\_Ratio\_List_k$ which is used to record the relative change improvement ratio (RCIR) for each CR_k in A as

$$CR\_Ratio\_List_k = CR\_Ratio\_List_k + \left(1 - \frac{\min\left(\left|f\left(u_i^{G+1}\right)\right|,\ \left|f\left(x_i^G\right)\right|\right)}{\max\left(\left|f\left(u_i^{G+1}\right)\right|,\ \left|f\left(x_i^G\right)\right|\right)}\right) \quad (94)$$

Equation (94) is implemented only when the $CR\_Flag\_List_i = 1$, which is the case where the trial vector is better than the target vector. The complete pseudocode of EADE is available in Ref. [85].

5.2.2.4. EFADE Algorithm

5.2.2.4.1. Advanced EFADE Mutation Scheme. EFADE proposed a novel mutation strategy coined as triangular mutation. Its main concept is to enhance the balance between the global exploration and local exploitation tendencies of DE by investing vectors with different properties in the same mutation rule. These vectors are as follows:

- x_{best}, x_{better} and x_{worst} are three random individuals selected from the current population ordered based on tournament selection.
- $\overline{x}^G_c$ is called a convex combination vector of the triangle (three randomly selected vectors combined in the form of linear equation) as:

Table 2
A summary of the type of mutation strategies used in 19 adaptive DE algorithms.

Algorithm	Mutation Scheme	Main Feature of the Scheme	Base Scheme
FADE	DE/rand/1	Fixed	DE/rand/1
jDE	DE/rand/1	Fixed	DE/rand/1
FiADE	DE/best/1	Fixed	DE/best/1
GADE	DE/rand/1	Fixed	DE/rand/1
DESAP	(non-DE standard scheme)	With deterministic mutation probability updates	Simple Perturbation Scheme
JADE wo	DE/current-to-*p*best/1	Utilization of the $100p\%$ best solutions	DE/current-to-best/1
JADE w archive	DE/current-to-*p*best/1 w archive	Utilization of the $100p\%$ best solutions and the A inferior solutions	DE/current-to-*p*best/1
DEGL	Local and Global Neighborhood-Based Mutations	Two stages mutation: Local and Global to balance the exploitation and exploration capabilities	DE/current-to-best/1
MDE_*p*BX	DE/current-to-gr_{best}/1	Utilization of the best individual selected from $q\%$ group of individuals	DE/current-to-*p*best/1
p-ADE	DE/rand-to-best/*p*best	Adaptive Dynamic Structure	PSO Standard Scheme [64]
SHADE	DE/current-to-*p*best/1 w archive	Utilization of the $100p\%$ best solutions and the A inferior solutions	DE/current-to-*p*best/1
L-SHADE	DE/current-to-*p*best/1 w archive	Utilization of the $100p\%$ best solutions and the A inferior solutions	DE/current-to-*p*best/1
$EsDE_r$-NR	DE/current-to-*p*best/1 w archive	Utilization of the $100p\%$ best solutions and the A inferior solutions	DE/current-to-*p*best/1
SaDE	Pool of standard DE strategies	Adaptive PCB among the schemes	DE/rand/1; DE/rand/2; DE/current-to-rand/1; DE/rand-to-best/1
EPSDE	Pool of standard DE strategies	Deterministic/Partial-predetermined	DE/best/1; DE/best/2; DE/rand-to-best/1; DE/rand-to-best/2; DE/rand/1; DE/rand/2; DE/current-to-rand/1
NRDE	Equally probable switch between two alternative mutation strategies	Deterministic/Partial-predetermined	DE/rand/1 and DE/best/1
SaDE-MMTS	Pool of DE/current-to-*p*best/1 w merged with two crossover schemes and no crossover	Adaptive PCB among the schemes	DE/current-to-*p*best/1 w
SaJADE	Pool of advanced DE strategies	Adaptive LPB among the schemes	DE/current-to-*p*best/1 wo; DE/rand-to-*p*best/1 wo; DE/current-to-*p*best/1 w; DE/rand-to-*p*best/1 w
EADE	Top, middle and bottom individual based mutation	Utilizes three different types of individuals in the same scheme. Top and bottom of $100p\%$ and middle of $Np - 2(100p\%)$. Deterministic/Partial-predetermined scheme is used to alternatively switch between the new scheme and DE/rand/1 scheme.	DE/rand/1
EFADE	Triangulation scheme with linear convex combination vector	Utilizes the best, better, and worst individual selected randomly from the current population. Deterministic/Partial-predetermined scheme is used to alternatively switch between the new scheme and DE/rand/1 scheme.	DE/rand/1

LPB: Learning Process Based **PCB**: Progressive-Controlled Based **EPB**: Evolution Process Based.

$$\overline{x}_c^G = w_1^* \cdot x_{best} + w_2^* \cdot x_{better} + w_3^* \cdot x_{worst} \quad (95)$$

where $w_i^* \geq 0$ and $\sum_{i=1}^{3} w_i^* = 1$; w_i^* are called the real weights and are calculated using $w_i^* = w_i / \sum_{i=1}^{3} w_i$, $i = 1, 2, 3$. w_i is calculated using the following equation:

$$w_i = 1 - \left| \frac{f(x_{best}) - f(x_i)}{f(x_{best}) - f(x_{max})} \right|, \quad i = 1, 2, 3 \quad (96)$$

As a minimization problem, $f(x_{best}) = f(x_{min}) = \min\{f(x_i)\}$, $i = 1, 2, 3$ is the individual with the minimum fitness value among the three randomly selected vectors. $f(x_{max})$ is the individual with maximum fitness value in the current population.

Using the vectors above, the new mutation scheme can be formulated as follows:

$$v_i^{G+1} = \overline{x}_c^G + F_1 \cdot \left(x_{best}^G - x_{better}^G\right) + F_2 \cdot \left(x_{best}^G - x_{worst}^G\right) + F_3 \cdot \left(x_{better}^G - x_{worst}^G\right) \quad (97)$$

where F_1, F_2 and F_3 are the mutation scalar factors that are generated in each generation using the new adaptation scheme. The mutation scheme above is implemented interchangeably with DE/rand/1 scheme with respect to a nonlinear probability condition as

$$If \left(rand(0,1) \geq \left(1 - \frac{G}{G_{\max}}\right)^2 \right) Then$$

$$v_i^{G+1} = \overline{x}_c^G + F_1 \cdot \left(x_{best}^G - x_{better}^G\right) + F_2 \cdot \left(x_{best}^G - x_{worst}^G\right) + F_3 \cdot \left(x_{better}^G - x_{worst}^G\right) \quad (98)$$

$$Else \quad v_i^{G+1} = x_{r1}^G + F \cdot \left(x_{r2}^G - x_{r3}^G\right)$$

where $rand(0,1)$ is a uniform random number generator between 0 and 1, $i \neq r1 \neq r2 \neq r3$ are indices randomly selected from the current population, and F is set to 0.7.

5.2.2.4.2. EFADE Parameter Control Schemes. EFADE proposes two adaptation schemes to update the values of F and CR during evolution. These new schemes have the advantage of not imposing extra parameters or determining a learning period while adapting the control parameters.

Table 3
This table encompasses taxonomy of the adaptation scheme used to update the main control parameters in 19 adaptive DE algorithms.

Algorithm Name	Parameter control strategies based taxonomy								
	Np		***F***		***CR***		***other control parameters***		
	Type of Setting	Type of Evidence	Type of Setting	Type of Evidence	Type of Setting	Type of Evidence	Parameter	Type of Setting	Type of Evidence
FADE	Tuned	×	Adaptive/PCB	Absolute	Adaptive/PCB	Absolute	*PC*; *FC*	Adaptive/PCB	Absolute
jDE	Tuned	×	Self-adaptive/EPB	Relative	Self-adaptive/EPB	Relative	×	×	×
FiADE	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	×	×	×
GADE	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	*PR*;*RI*;CR_m; μ_{CR}	Adaptive/LPB	Relative
DESAP	Self-adaptive/EPB	Relative	Tuned	×	Self-adaptive/EPB	Relative	η	Self-adaptive/EPB	Relative
JADE wo	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	μ_F; μ_{CR}	Adaptive/LPB	Relative
JADE w	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	μ_F; μ_{CR}	Adaptive/LPB	Relative
DEGL	Tuned	×	Tuned	×	Tuned	×	α;β; k	Tuned	×
							w	Deterministic/Partial-predetermined; self-adaptive/EPB	Absolute and Relative
MDE_*p*BX	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	μ_F; μ_{CR}	Adaptive/LPB	Relative
							w_F; w_{CR}	Deterministic/Partial-predetermined	Absolute
p-ADE	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	W; K	Adaptive/LPB	Relative
SHADE	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	M_{CR}; M_F;w_k	Adaptive/LPB	Relative
							p	Deterministic/Partial-predetermined	Absolute
L-SHADE	Adaptive/PCB	Absolute	Adaptive/LPB	Relative	Adaptive/LPB	Relative	M_{CR};M_F;w_k	Adaptive/LPB	Relative
							p	Deterministic/Partial-predetermined	Absolute
$EsDE_r$-NR	Adaptive/PCB	Absolute	Adaptive/LPB	Relative	Adaptive/LPB	Relative	$freq_i$; μ_{freq}; M_F; M_{CR}; w_k	Adaptive/LPB	Relative
							y_i; σ	Deterministic/Partial-predetermined	Absolute
SaDE	Tuned	×	Deterministic/Partial-predetermined	Absolute	Adaptive/PCB	Absolute	p;S	Adaptive/PCB	Absolute
EPSDE	Tuned	×	Adaptive/PCB	Absolute	Adaptive/PCB	Absolute	×	×	×
NRDE	Tuned	×	Deterministic/Partial-predetermined	Absolute	Deterministic/Partial-predetermined	Absolute	δ	Adaptive/PCB	Absolute
SaDE-MMTS	Tuned	×	Deterministic/Partial-predetermined	Absolute	Adaptive/PCB	Absolute	p;S	Adaptive/PCB	Absolute
SaJADE	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	μ_F; μ_{CR}; μ_S;S	Adaptive/LPB	Relative
EADE	Tuned	×	Deterministic/Partial-predetermined	Absolute	Adaptive/LPB	Relative	*CR_Ratio_List*	Adaptive/LPB	Relative
EFADE	Tuned	×	Adaptive/LPB	Relative	Adaptive/LPB	Relative	p;ps; s	Adaptive/LPB	Relative

LPB: Learning Process Based **PCB**: Progressive-Controlled Based **EPB**: Evolution Process Based.

Scalar factor adaptive scheme: The basic concept of this adaptation scheme is how to dynamically maintain a proper balance between the local exploitation and global exploration aspects during generations. For this purpose, three different scalar factors (F_1, F_2 and F_3) have been involved in the new mutation scheme. The values of these parameters are updated in each generation as follows:

$$F_i = rand(0,\ k_i), \quad i = 1, 2, 3 \tag{99}$$

where

$$k_1 = \begin{cases} \left|\frac{f(x_{better})}{f(x_{best})}\right| + \varepsilon, & if\ \left|\frac{f(x_{better})}{f(x_{best})}\right| < 1 \\ \left|\frac{f(x_{best})}{f(x_{better})}\right| + \varepsilon, & otherwise, \end{cases}$$

$$k_2 = \begin{cases} \left|\frac{f(x_{worst})}{f(x_{best})}\right| + \varepsilon, & if\ \left|\frac{f(x_{worst})}{f(x_{best})}\right| < 1 \\ \left|\frac{f(x_{best})}{f(x_{worst})}\right| + \varepsilon, & otherwise, \end{cases} \tag{100}$$

$$k_3 = \begin{cases} \left|\frac{f(x_{worst})}{f(x_{better})}\right| + \varepsilon, & if\ \left|\frac{f(x_{worst})}{f(x_{better})}\right| < 1 \\ \left|\frac{f(x_{better})}{f(x_{worst})}\right| + \varepsilon, & otherwise, \end{cases}$$

Table 4
DE algorithms points of strengths based on mutation strategy.

Algorithm	Points of strengths of the DE mutations
FADE; jDE	The standard DE/rand/1(Eq. (2)) scheme is always considered as the fastest non greedy scheme with good convergence performance [20].
JADE	The JADE mutation (Eq. (34)) alleviates the problem of premature convergence because of its ability to intensify the population diversity [149].
JADE with archive; SHADE; L-SHADE; EsDE$_r$-NR	• Increase the population diversity as such the problem stem from the convergence rate was further reduced. This is so because, the superior and inferior solutions are both incorporated into the mutation strategy. • No significant computational overhead as the archive operation has been made very simple. • SHADE and L-SHADE over JADE algorithm have the advantages that they dynamically update the value of p top solutions using a deterministic based technique. The above points are discussed in Ref. [9,124,125,149] using Eq. (35). The literature is already rich with other analysis and modified versions of SHADE algorithm as this algorithm proved effectiveness in its performance [132].
DEGL	The local and global neighborhood topology (ring) in the new DE mutation (Eq. (44)) creates the proper balance between the exploitation and exploration capabilities of the algorithm during evolution [31].
MDE_*p*BX	It weaknesses the tendency of premature convergence and alleviates the attraction of any fixed point in the fitness landscape. This modification in (Eq. (48)) has led to reduce the greediness feature of the DE mutation scheme towards choosing the superior solutions for perturbation and making it converges fast to a local point [59].
p-ADE	The ability to be dynamically changed to three different schemes in (Eq. (56)) according to the classification conditions upon the individuals' quality that are located in the same population [15].
SaDE; SaDE-MMTS; SaJADE	Learning strategies (Eq. (82) and Eq. (89)) have been applied to gradually evolve the selection of one mutation scheme from a pool of mutation schemes in hand throughout generations. This characteristic allows further improvements in the DE moves, thus increasing the exploitation features of the algorithm. This learning strategy affords flexibility to be extended to include more candidate mutation schemes in an attempt to solve complex optimization problems [54], [104, 151].
EPSDE	The selection of the mutation strategies is made randomly from a pool of mutations with different characteristics to avoid the influence of less effective mutation strategies [79].
NRDE	The mutation strategy adopted in Ref. [53] is simple but effective. It alternatively switches between two extreme mutation strategies (DE/rand/1 Eq. (2) and DE/best/1 Eq. (3)). The use of both strategies will enhance the search process as each scheme has its capability to either provide high explorative or high exploitative depending on the nature of strategy.
EADE	It (Eq. (93)) successfully maintains the global exploration and local exploitation of the search process through investing components from the best, middle and worst vectors in DE population [85].
EFADE	A novel mutation scheme named as triangulation mutation (Eq. (97) and (98)) based on convex combination vector has been proposed to successfully enhance the global and local search capabilities. This new scheme sorts three randomly chosen vectors into best, better and worst and employs them in the weighted convex equation and the main mutation scheme. The different components inherited from these vectors have helped to archive the proper balance between the two contradictory aspects of the search process [89].

Table 5
DE algorithms drawbacks based on mutation strategy.

Algorithm	Drawbacks of the DE mutation
FADE; jDE	**Lack of Population diversity** This problem of Eq. (2) will arise in optimizing high dimensional functions and also when the characteristic of the test problem is challenging [20] and [74].
DESAP	**Lack of Exploitation and Exploration Ability** No significant improvements over the standard DE. The new crossover and mutation operations (Eq. (20) and (21)) are simple and straightforward schemes, they did not bring about the desired performance and DESAP outperform the standard DE in only one out of five test problems [126].
JADE; JADE with archive; SaDE-MMTS; SaJADE; SHADE; L-SHADE; EsDE$_r$-NR	**Stagnation** The population selective factor, p in Eq. (34)–(36) is tuned before the run and kept fixed during the evolution process. This strategy might lead to stagnation problem especially after several epochs of evolution when the population diversity rate is very low and the superior solutions in the p% of the current population start to be close in values if not exactly same [54], [149,151]. Although, SHADE and L-SHADE suggested using deterministic based technique to update the value of p as in Eq. (70), it still has the aforementioned problem may occur at the late stages of evolution [124], [125].
DEGL	**Limited test suit functions have been used to test the effectiveness of DEGL.** This implies that DEGL may not be able to model real-world complex problems [31].
MDE_*p*BX	**Lack of strategy and parameter analysis & Crossover greediness tendency** • The influence of the new mutation scheme in Eq. (48) on the diversity of population and convergence rate is not investigated. • The greediness of the new crossover scheme towards superior solutions even with the associated dynamic parameter, p (see Eq. (55)) of the top-ranking vectors may lead to premature convergence problem. The above two points are discussed and investigated in Ref. [59].
p-ADE	**Local optimum caused by greedy mutation** Selecting the best solutions from the previous and current population may lead the new strategy in Eq. (56) become greedier towards good solutions, thus running the risk of falling into local optimum [15].
NRDE	**The use of equibrobale switch between the two mutation strategies is challenging** The two mutation strategies adopted (Eq. (2) and Eq. (3)) are simple and one of them is greedy strategy. This may drift the search towards falling into local optima [53].

where $rand(0,\ k_i)$ is a uniform random generator between 0 and k_i, and $\varepsilon = 0.0001$ is a small number added to avoid zero value which implies no perturbation. If $k_i > 1$, then its value is truncated to 1. This adaptation scheme of F_1, F_2 and F_3 have successfully achieved the desired balance of the two contradictory aspects by controlling the amplification of the perturbation of the mutation scheme through generations.

Crossover rate adaptive scheme: The main idea of this scheme is to select an appropriate control value for the *CR* of each target vector from recommended setting values in Ref. [110], distributed into two sets. Then, the selection of these values is updated using learning strategy, depending on their experiences of producing promising vectors through generations as follows:

$$\begin{aligned} &If\ G = 1\ \ then \\ &\quad CR_i^1 = \begin{cases} CR_1, & if\ \ rand(0,1) \le 0.5 \\ CR_2, & otherwise \end{cases} \\ Else \quad &\quad CR_i^G = \begin{cases} CR_1, & if\ \ rand(0,1) \le\ \ p_1 \\ CR_2, & if\ \ p_1 < rand(0,1) \le\ p_1 + p_2 \end{cases} \end{aligned} \tag{101}$$

where $CR_1 \in [0.05, 0.15]$ and $CR_2 \in [0.9,\ 1]$ are the two suggested sets, p_j, $j = 1, 2, \ldots, m$ is the probability of selecting the *CR* value from set j which is first initialized to $1/j$, and $m = 2$ is the total number of sets. p_j is dynamically updated during generations using the following equation:

Table 6
DE algorithms points of strengths based on parameter control schemes.

Algorithm	Points of strengths of parameter control schemes
FADE	• The first attempt in using Fuzzy Control in DE parameter settings for the sake of reducing the user load from parameter tuning. • Possess robustness and fuzziness with problems in an imprecise environment. As discussed and presented in Ref. [74].
jDE	Adjust both *F* (see Eq. (9)) and *CR* (see Eq. (10)) in a self-adaptive manner with few additional parameters [20].
FiADE	The adaptation schemes of *F* (see Eq. (11)–(13)) and *CR* (see Eq. (14)) are very simple and yet effective. The best feature of these schemes is their strategy to work alternatively as the evolution process requires [52].
GADE	The greedy adaptation procedure for adjusting the *F* and *CR* values used in GADE repeatedly finds the better parameter assignments by searching in the neighborhood area of the current assignments. It has two significant advantages [70]. • It offers a good chance for the better trial solutions to survive. • It sustains the improvement rate of the fitness values in the next generations.
DESAP	• The first attempt to demonstrate the possibility to produce an adaptive algorithm that not only updates the crossover (see Eq. (24) and (25)) and mutation rates (see Eq. (26) and (27)) but also the population size parameter as well as in Eq. (22) and (23). • Downsize DE parameters' setting by updating the population size and other control parameters in an adaptive manner. As discussed and presented in Ref. [126].
JADE; JADE with archive; SaJADE	• Adjusting the values of *F* and *CR* in an adaptive characteristic based on a learning strategy that gains knowledge from previous iterations and cumulates it into two learning parameters, μ_F and μ_{CR} to be retained and used in the current population (see Eq. (37)–(41)). • Create the proper balance in maintaining the pressure on the population to move towards exploring more optimal solutions, as well as not to lose the exploitation features [54,149].
DEGL	The scalar factor *w* (see Eq. (45)–(47)) has been used to control the behavior of the mutation during evolution. It helps to provide the required balance between the local and global moves in the search space [31].
MDE_*p*BX	Modifies the original JADE scheme of adapting the values of *F* and *CR*. The new modifications to the control parameters schemes with the combination of the new mutation and crossover schemes (see Eq. (49)–(55)) in MDE_*p*BX have greatly increased the ability of exploitation and exploration, and the search process is directed to explore better search space regions, hence, escaping from the possibility of getting trapped in suboptimal solutions [59].
p-ADE	• Possess a unique merit over other adaptive DE algorithms by involving the number of iterations passed over and the fitness value in the updating process; for the sake of accelerating the convergence rate [15]. • Creating the required balance between the exploitation and exploration features. This has been achieved through selecting the best values for the control parameters *W*, *K*, *F*, and *CR* as expressed in Eq. (59)–(62).
SHADE; L-SHADE	• The former adaptation schemes of JADE for *F* and *CR* have been improved by applying a new dynamic scheme based on accumulating the successful parameter control settings in a historical memory updated during evolution (see Eq. (63)–(70) and Eq.(72)–(75)). This strategy has the advantage of guiding the search process towards better solutions as the negative impact caused by the poor setting values that may have been included in S_{CR} and S_F is resolved [124,125]. • L-SHADE continually decreases the population size using a linear reduction equation that positively reduces the computation cost of the algorithm (see Eq. (71)) as presented in Ref. [125].
$EsDE_r$-NR	• The two sinusoidal adjustments (increasing and decreasing) have the advantage of updating the value of *F* in such a way that drifts the search direction to different regions as expressed in Eq. (76)–(78). • The niche-based mechanism suggested to adapt the *Np* (see Eq. (81)) value has the advantage of overcome the problem of stagnation that the algorithm is probably falls in when reducing the population size. • The restart method (see Eq. (79) and (80)) has the advantage of enriching the later generations of the search with new solutions. The aforementioned three points on EsDEr-NR performance is discussed in Ref. [9].
SaDE; SaDE-MMTS	Utilizes an adaptive learning strategy to adjust the crossover rate (see Eq. (85)) and another learning scheme to adaptively select mutation strategy during the search process as provided in Eq. (82) [104,151].
EPSDE	The selection of the mutation and crossover parameter values is made randomly from a pool of values to avoid the influence of less effective parameter settings [79].
NRDE	• The adaptation mechanism used to modify the values of *F* and *CR* is simple. It requires no knowledge from the search space or feedback. No means or variance, etc or any other parametric probability distribution parameters are involved. It only needs to know the feasible ranges of these parameters [53]. • The additional control parameter δ (see Eq. (87) and (88)) has a significant effect on the performance of the new proposed DE selection scheme. The use of this self adaptive parameter in conjunction with the new selection scheme will improve the search process from getting trapped into a basin of optima attraction. This is practically achieved by including inferior solutions occasionally survived using the new selection scheme [53].
EADE	The crossover rate has been modified during the run using an adaptive rule (see Eq. (94)) that collects information from past experiences of successful and unsuccessful individuals during and after a learning period [85].
EFADE	No extra parameters or learning period burdens. The adaptive process is implemented using simple and clear schemes for updating the values of *F* and *CR* as expressed in Eq. (99)–(104). Both schemes show advantages to balance the local exploitation and global exploration capabilities of the algorithm [89].

$$p_j^{G+1} = \left((G-1)\cdot p_j^{G-1} + ps_j^G\right)\Big/ G, \tag{102}$$

$$ps_j^G = \frac{s_j^G}{\sum_{j=1}^{m} s_j^G}, \tag{103}$$

where

$$s_j^G = \frac{ns_j^G}{\sum_{g=1}^{G} ns_j^g + \sum_{g=1}^{G} nf_j^g} + \varepsilon, \tag{104}$$

where ps_j^G is the probability of selecting the j^{th} set, s_j^G is the success ratio of the individuals who successfully survived to the next generation, ns_j^G and nf_j^G are two counters for the individuals who survived or failed in the selection operations, respectively, and $\varepsilon = 0.01$ is a very small number to avoid the zero value of s_j^G. Arguably, the small values of CR have exert effects at the early stage of evolution because of the high diversity the first populations always have. By contrast, in the final generations, the large values of CR can increase the diversity of population as the individuals will all be close to the best individual.

5.3. Discussion on adaptive DE variants

Nineteen adaptive DE algorithms have been presented in the previous subsections. These algorithms clearly exhibit diversity in terms of characteristic, structure, complexity and algorithmic logic. Despite their advantages, these algorithms show shortcomings in some particular cases. In this subsection, comparisons have been established among these methods according to the taxonomy provided in Section 3, their algorithmic design similarities and differences and pros and cons of each

Table 7
DE algorithms drawbacks based on parameter control schemes.

Algorithm	Drawbacks of DE parameter control schemes
FADE	**Lack of algorithm performance analysis** It does not involve any relative consideration for the individual fitness value only absolute based on the knowledge gained from the fuzzy controller [74].
jDE	**Lack of balance between the exploitation and exploration** There is a weakness in the relative consideration of the individual fitness value since the values of F and CR are selected according to the best fitness picked up from the current generation only, then updated according to a uniform distribution (see Eqs. (9) and (10)). This may lead jDE to be biased towards exploration [20].
FiADE	**Limited test-suit** Its simplicity may lead FiADE to fail to model very complex optimization problems [52].
GADE	**Greedy adaptive scheme** The greediness tendency of the new adaptive scheme in GADE towards the parameter assignments that produce the best solutions may lead to a stagnation problem in DE [70].
DESAP	**Lack of population diversity** • It outperforms the standard DE over five test functions only. • It was found that both DESAP's versions yielded highly similar results in terms of the best solution obtained; although, in providing more stability DESAP with absolute encoding was more favorable than DESAP with relative encoding. This can be observed from the experiments provided in Ref. [126].
JADE; JADE with archive; SaJADE	**Problem-dependent parameter selection** The parameters c and p determine the adaptation rate of μ_{CR} and μ_F and the greediness of the mutation strategy are kept fixed during the run (see Eq. (34)–(36)). These two parameters have the ability to affect the overall performance of JADE mutation [124,149].
JADE with archive; SHADE; L-SHADE	**Inappropriate parameters update** The adaptive schemes of F and CR depend on the constraint $f(u_i) < f(x_i)$ because of the weighted scores associated with these schemes; so, when $f(u_i) = f(x_i)$ the weight becomes 0 resulting in inappropriate parameter settings [124].
MDE_*p*BX	**Lack of theoretical guidelines** There are two additional control parameters q (the group size in the mutation operation in Eq. (48)) and p (the number of the top-ranking vectors in the crossover operation in Eq. (55)). These parameters bring about the effect to the performance of the mutation and crossover. There is lack on the theoretical evidence of the importance of these two control parameters [59].
p-ADE	**Time consumption** All of the four parameters (W, K, F, and CR) should be adjusted through the run (see Eq. (59)–(62)), simultaneously with adjusting the mutation scheme (see Eq. (56)–(58)). This may require increasing the number of iterations needed to achieve the optimal solution [15].
SaDE; SaDE-MMTS	**Deterministic evidence rule for updating the mutation factor** • The mutation factor, F is updated through deterministic rule based, though, it has been set to different random values through the evolution process (see Eq. (83)). • It introduces additional learning parameters such as ns and nf to steer both learning strategies, thus making the algorithm cumbersome with many parameters (see Eq. (82)). This can be clearly observed in Ref. [104].
EPSDE	**Local optima** The procedure of adjusting the control parameter values is mostly implemented in a random way. There is no accumulating knowledge through the evolution and the parameter value that produce inferior solution will be reinitialized at the same generation. This procedure in conjunction with the random selection of the mutation scheme will in some cases divert the population to a wrong direction of the search space and fall in local optima if the algorithm fails to select the appropriate parameter values and scheme [79].
NRDE	**Deterministic evidence rule for updating the mutation factor and crossover rate** Despite the fact that the adaptation strategy of F and CR is simple and effective [53], it is arguably known that the search through different problems landscapes requires different parameter settings during the run and getting feedback from the search will always be necessary [63].
EADE	**Lack of exploitation** In Ref. [85], it can be noted that, • The scalar factor has been set to a fixed value during the run; this may affect the exploitation ability of the mutation scheme and reduces the chances of finding good solutions around the local regions of the target individual. • Although the adaptation scheme of CR is quite simple, it has disposes many parameters to the algorithm.

Table 8
Summary of 9 important and recent adaptive DE variants.

Algorithm	Ref	Mutation Strategy	Parameter Control Schemes
Composite DE (CoDE)	[135]	Ensemble of DE mutation strategies "DE/rand/1bin", "DE/rand/2/bin" and "DE/current-to-best/bin" is created. One is chosen randomly for each individual and the best will survive for the next population.	Combinations of F and CR values are created as $[F = 1.0, CR = 0.1]$, $[F = 1.0, CR = 0.9]$ and $[F = 0.8, CR = 0.2]$. At each generation, each individual is assigned to randomly selected combination and the best will survive to the next generation.
Harmony search based parameter ensemble adaptation for DE (HSPEADE)	[78]	This algorithm is an improvement of the EPSDE algorithm. In this algorithm the Harmony Search algorithm (HS) proposed in Ref. [50] is employed to select the DE strategies (Eq. (35) and Eq. (48)) from an ensemble of these strategies instead of random manner as in EPSDE.	It is a harmony based parameter selection approach used to select the values of F and CR on the basis of Harmony search technique. In which a Harmony memory is used to store the successful strategies and parameter control values to be used in the next generations.
Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood (LSHADE-*cn*EpSin)	[10]	• The JADE current/to-pbest (Eq. (35)) mutation strategy is adopted in this algorithm. • The crossover operator is modified using a covariance matrix learning $C = BDB^T$ with Euclidean neighborhood between the best individual and every other individual in the population; where, B and B^T are two orthogonal matrices and D is a diagonal matrix which contains the Eigen values.	• The value of Np is adapted during evolution using the linear population size reduction technique proposed in LSHADE-EpSin algorithm [11]. • The value of F is adapted during the run using an ensemble of two adaptation schemes (sinusoidal approaches): adaptive sinusoidal increasing adjustment and non-adaptive sinusoidal decreasing adjustment (Eq. (76)–(78)). • The value of CR is adapted in all generations using normal distribution as in L-SHADE (Eq. (72)–(75)).
Differential crossover strategy based on covariance matrix learning with Euclidean neighborhood (L-*covn*SHADE)	[12]	• The JADE current/to-pbest mutation strategy (Eq. (35)) is adopted in this algorithm. • The crossover operator is modified using a covariance matrix learning $C = BD^2B^T$ with Euclidean neighborhood between the best individual and every other individual in the population; where, B and B^T are two orthogonal matrices and D^2 is a diagonal matrix which contains the Eigen values.	• The value of Np is adapted during evolution using the linear population size reduction technique proposed in L-SHADE algorithm (Eq. (71)). • The value of F is adapted using Cauchy distribution and CR value is adapted using normal distribution along with their corresponding formulas as suggested in L-SHADE algorithm (Eq. (63), Eq. (72) and Eq. (73)–(75)).
Individual-dependent differential evolution with population size adaptation (IDEbd)	[23]	• New mutation strategy has been proposed: $u_i = \begin{cases} x_o + F_o \cdot (x_{r1} - x_o) + F_o \cdot (x_{r2} - x_{r3}) & \text{if } o \in S \\ x_o + F_o \cdot (x_{better} - x_o) + F_o \cdot (x_{r2} - x_{r3}) & \text{if } o \in I \end{cases}$ Where o is the base individual's index; S is the set of superior and inferior individuals in the population. • Another perturbation scheme is proposed for the last individual with small probability to enhance the local search before implementing the binomial scheme.	• The Np value is adapted during the run from the maximum value which is 1 to the minimum value which is close to 0 depending on the population diversity measure which is named as relative diversity: $rdiv = \frac{div}{div_{init}}$ Where div is the current population diversity and div_{init} is the diversity of the initial population calculated using the diversity equation proposed in Ref. [99]. • Both F and CR values are adapted using normal distribution with distribution means o/Np and i/Np, respectively.
LSHADE with Semi-parameter adaptation hybrid with CMA-ES (LSHADE-SPA)	[86]	The JADE current/to-*p*best mutation strategy (Eq. (35)) is adopted in this algorithm.	• The value of Np is adapted during evolution using the linear population size reduction technique proposed in L-SHADE algorithm (Eq. (71)). • The values of F and CR are adapted using two semi parameter adaptation (SPA) parts: **Part1:** during this part the concentrate will be on CR which is updated using L-SHADE (Eq. (72)–(75)); whereas F is updated using uniform distribution within specified limits. **Part2:** during this part the value of F is updated using L-SHADE (Eq. (63) and Eq. (73)–(75)); whereas, the value of CR is updated using the same scheme used in part1. The first part is activated during the first half of evolution. The second part is

(*continued on next page*)

Table 8 (*continued*)

Algorithm	Ref	Mutation Strategy	Parameter Control Schemes
			activated during the second half of evolution.
DE with automatic adaptation of operators and control parameters (DE-AOPS)	[40]	A set of DE strategies SO_{set} = {Eq. (35) and [112]} is created and a suitable DE operator is selected with regard to the better performing strategy in each stage of the evolution process. This is implemented using rank strategy for the DE operators involved in the search process: $Rank_p = \frac{SO_{p.suc}}{NI_{op}}$ Where NI_{op} is the number of individuals generated using DE operator SO_{op}.	The values of F and CR are generated from a set of fixed values between 0 and 1 as combinations. The better combinations of F and CR values are survived on the basis of ranking method similar to that of the operators selection: $Rank_y = \frac{com_{y.suc}}{NI_y}$ Where NI_y is the number of individuals influenced by com_y.
Ensemble of multiple DE variants (EDEV)	[139]	Multi-population based framework (MPF) for ensemble of multiple DE variants is considered: $pop = \underset{i=1\ldots4}{\cup} pop_i$ Where pop_1, pop_2, pop_3 are three indicator subpopulations from three different variants JADEw, CoDE and EPSDE respectively, and pop_4 as reward subpopulation. pop is the overall population.	• All indicators have equal population size and they are very much smaller than the reward indicator subpopulation. Thus the Np formula will be $Np_i = \lambda_i \cdot Np$ where Np is the pop size and Np_i is the size of pop_i. λ_i is the proportion between pop_i and pop. As such, we have $\sum_{i=1\ldots4} \lambda_i = 1$. In EDEV, $\lambda_1 = \lambda_2 = \lambda_3$. • F and CR are adapted based on JADEw, CoDE and EPSDE each within its representative indicator subpopulation.
Adaptive guided DE (AGDE)	[87]	A novel mutation rule has been proposed to maintain the balance between the global exploration and the local exploitation as well as to improve the convergence rate of DE, as follows: $v_i^{G+1} = x_r^G + F \cdot (x_{p\_best}^G - x_{p\_worst}^G)$ where $x_{p\_best}^G$ and $x_{p\_worst}^G$ are randomly chosen from the top and bottom 100p% vectors, respectively in P^G, x_r^G is randomly chosen from the middle $[Np - 2(100p\%)]$ vectors, and $p \in (0\%, 50\%)$.	• F are independently generated according to uniform distribution in $(0.1, 1)$. • CR are generated with accordance to the new adaptation scheme explained in Eq. (101)–(104).

algorithm are provided in Tables 2–7 In addition to these 19 DE variants, another 9 important and recent DE algorithms have been summarized in Table 8 to offer more adaptive algorithmic design insights. Finally, an estimated rank of these algorithms has been presented in Fig. 4.

5.3.1. Adaptive DE conceptual similarities and differences

In this subsection, we discuss how the aforementioned methods relate and differ from one another and from the standard DE algorithm based on mutation strategy and parameter control schemes used in each algorithm.

5.3.1.1. Comparison-based DE mutation scheme. The 19 algorithms presented under this comparison differ in their mutation strategies, as seen in Table 2, which summaries the mutation schemes employed in each algorithm and categorized according to the taxonomy provided in Section 3. From Table 2, we can observe that the most common mutation strategy is DE/current-to-*p*best/1 with and without archive. This new strategy has been adopted in many DE algorithms such as JADE, SHADE, L-SHADE and EsDE$_r$-NR algorithms because this strategy has proved effective performance. All the algorithms adopt the classical crossover (*bin*) and the standard selection operations in their main work except DESAP, which uses a modified crossover scheme similar to that of DE/rand/1 mutation scheme, MDE_*p*BX, which uses a new modified crossover scheme called *p*-Best, and NRDE that uses a simple mechanism to switch between two crossover schemes, one of these schemes is newly proposed in NRDE; while SaDE, EPSDE, NRDE, SaDE-MMTS, SaJADE, EADE and EFADE invent new adaptive schemes that can make a selection among a pool of candidate mutation schemes. In addition, SaDE-MMTS uses three crossover aspects (binomial, exponential and no crossover) integrated with their corresponding mutation strategies creating a pool of DE strategies to automatically adapt one of them.

5.3.1.2. Comparison-based DE parameter control schemes. The comparison in this subsection is based on the parameter setting taxonomy presented in Section 3. Table 3 elucidates the main features of the adaptation scheme used to control the three main control parameters (*F*, *CR and Np*) in the 19 adaptive DE variants. This table shows how the adaptive concept is revealed by applying an adaptive rule to at least one of the algorithm components. From the same table, none of the algorithms has considered adapting the population size *Np* except DESAP, L-SHADE and EsDE$_r$-NR algorithms. The focus is only on *F* and *CR*.

5.3.2. Adaptive DE strengths and drawbacks

In this subsection, conceptual strengths and drawbacks of the 19 algorithms have been discussed in terms of modifying the main DE strategies, additional components that have been included in the original DE algorithm and function as adaptive features. All the modifications and integrations proposed include extra moves in the original DE as well as create the proper balance between the exploitation and exploration characteristics. However, drawbacks need to be considered.

5.3.2.1. Comparison-based DE mutation strategy. In this subsection, the 19 DE algorithms have been compared based on the mutation scheme used in each algorithm. Tables 4 and 5 illustrate the comparison of algorithms in terms of pros and cons of each mutation strategy.

5.3.2.2. Comparison-based DE parameter control schemes. Tables 6 and 7 elucidate the points of strengths and drawbacks of the adaptation scheme used to control the three main parameters (*F*, *CR* and *Np*) in the 19 DE variants. The two tables indicate common pros and cons among certain algorithms because they use the same adaptation scheme(s).

5.3.3. Supplemental adaptive DE variants

Table 8 gives a summary of 9 important and very recent adaptive DE algorithms each according to its mutation strategy or ensemble of strategies it utilizes and the adaptive scheme(s) it employs to adjust the values of *Np*, *F* and *CR* control parameters. This table gives us an indication that the L-SHADE's adaptive schemes have dominated over many other adaptive

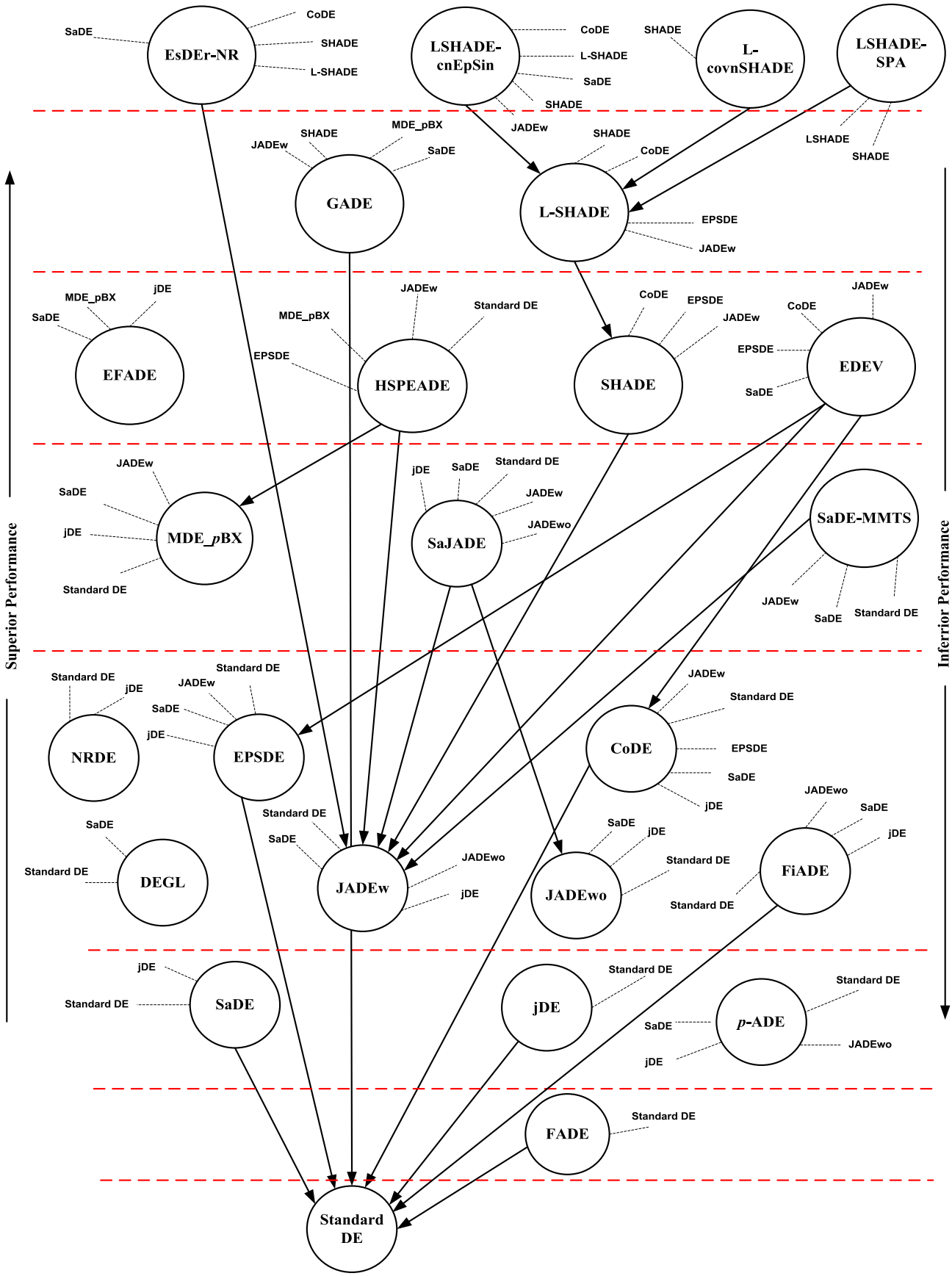

Fig. 4. An estimated rank of the adaptive DE algorithms.

Table 9
Future insights suggested for more DE algorithmic design investigation.

Suggestion	Details
Multi-comparison statistical test	It would be interesting to use some multi-comparison statistical test such as Friedman test, ANOVA and Wilcoxon Rank to analyze the differences among the state-of-the-art adaptive DE variants presented in this survey.
Improve the performance of the JADE mutation strategy and its variants [149]	The selection of the best individuals, p% of the population size in the mutation strategy can be implemented in an adaptive manner based on the population diversity.
Empirical and theoretical improvements in DEGL [31]	The neighborhood size in DEGL can be selected based on empirical guidelines for different optimization problems. Another suggestion is to study the performance of DEGL using different neighborhood topologies. These topologies can be, e.g. wheel shaped or star-shaped topologies.
Improve the performance of the MDE_pBX algorithm in different directions [59]	The MDE_pBX algorithm is a platform for many modifications. 1) An analytical investigation on the effects of the two new strategies (mutation and crossover) on the population diversity and convergence rate. 2) The connotation of a dynamic grouping can be a future MDE_pBX development to include new operators such as *DE/gr_best/*1, *DE/gr_best/*2, etc., and then their effectiveness could be measured on different types of test functions. 3) The parameter p, may also be modified to be adaptive or at the very least dynamic during the evolution process, hence its performance effectiveness can further be investigated. 4) There are two additional control parameters q (the group size in the mutation operation) and p (the number of the top-ranking vectors in the crossover operation), a theoretical guidelines of how to select the values of p and q can be investigated.
Many future insights for EADE [85]	1) Propose an adaptive mechanism for F and test the new algorithm on different optimization problems. 2) Integrate different adaptive schemes for CR and compare the results with different adaptive DE algorithms. 3) Investigate the performance of EADE in solving complex problems such as constrained and multi-objective optimization problems. 4) Combine the novel mutation strategy with other EAs such as genetic algorithm and particle swarm optimization.
Interesting improvement for EFADE [89]	It would be combining the new algorithm with other DE mutation schemes (standard or advanced). Another future direction would be integrating the new triangulation scheme with other adaptive DE algorithms such as SHADE. Finally, an experimental study can be conducted by increasing the number of CR sets and study their effects on different optimization problems.
SHADE and L-SHADE have the same future work development [125]	In which an adequate adaptive technique can be found to dynamically adjust the memory size parameter for new types of problems.
The neighborhood steps in GADE can be adjusted dynamically [70]	This proposed idea will have a great effect on enhancing the exploration and exploitation capabilities of the GADE algorithm.
The integration of EsDEr-NR with other DE variants [9]	The significant adaptation strategies used in EsDE$_r$-NR can be integrated with other DE variants and the results obtained can be examined.
Enhance the adaptive scheme of the parameters control in the SaDE and its variant such as SaDE-MMTS [104]	In these two algorithms, the parameter F can be set to an adaptive rule that accumulate knowledge from the previous generations.
Improve the adaptive ensemble of EPSDE [79]	The random strategy of the EPSDE in selecting the parameters control and mutation strategies can be improved by accumulating knowledge regarding the performance of the control parameter values through certain number of generations.
Integrate the adaptation schemes of ***F*** and ***CR*** in FiADE with other State-of-the-art DE algorithms [52]	This may lead to more competitive adaptive DE variants if these adaptation schemes of FiADE are integrated with other non adaptive DEs, e.g. opposition-based DE [106]. This step may lead to further future insight if a theoretical study is investigated on how the values of F and CR are adapted in the new proposed algorithms.
NRDE has many open issues as future work insights [53]	The switching mechanism of the strategy and parameter control can be further investigated in other complex problems in addition to noisy functions. The dynamic switching of the DE parameters can also be analytically investigated.

schemes employed in DE. Likewise, the JADE mutation strategy with archive (see Eq. (35)) has been adopted in many adaptive DE algorithms.

5.3.4. Adaptive DE schema ranking

Fig. 4 depicts an estimated rank of 24 adaptive DE algorithms based on their average performance in comparison with state-of-the-art DE algorithms (as presented in literature). In this figure, the sold arrow indicates that the algorithm in the source node has inherited the mutation strategy from the destination node; whereas, the intermittent line refers to the DE variants with less or almost equal performance in comparison with the DE algorithm in the source node. We can also observe that Fig. 4 is divided into distinct layers. Each layer indicates that the DE algorithms placed in the same layer have almost equal performance.

Generally, Fig. 4 shows that the adaptive DE algorithms with advanced mutation strategies such as JADE with archive [149], MDE_pBX [59] and DEGL [31] have gained the overall best performance. This is because these advanced strategies could handle the deficiencies in the standard DE strategies such as the greediness in the DE/best/1 strategy and successfully managed to create the required balance between the exploration (i.e. the global search tendency) and exploitation (i.e. the local search tendency) capabilities of the algorithm. Therefore, we can observe from the same figure that the DE algorithms placed in the top layers such as SHADE [124] have already adopted these advanced strategies (e.g. DE/current-to-pBest/1 strategy in Eq. (35)) and incorporate them with the parameter adaptive schemes to achieve the best algorithmic design performance. Likewise, adopting effective algorithms' schemes also applies to the parameter control schemes. LSHADE-*cn*EpSin [10], L-*covn*SHADE [12] and LSHADE-SPA [86] algorithms are already a modified versions of L-SHADE algorithm' parameter control schemes [125]. Another important type of adaptive DE variants is the dynamic selection of multiple DE strategies during the evolution process such as in EPSDE [79], HSPEADE [78] and CoDE [135]. This type of adaptive algorithms has always show good performance and still a very hectic research trend; because, it has already been proved through experiments that at each stage of evolution there is a suitable DE strategy to impose different search step size and guide the search towards better direction. As such, EDEV algorithm [139] outperforms JADE with archive, CoDE, SaDE and EPSDE algorithms with outstanding performance as this algorithm proposed an adaptive scheme that manages to select the suitable adaptive DE scheme during the run.

6. Metaheuristic framework with parameter control

After an interpretation of the literature and a comparative analysis of the aforementioned algorithms' design, we offer some guidelines on how to design and implement adaptive DE algorithms. The proposed designing framework provides the reader with the main steps that are required for integrating any proposed meta-algorithm with parameter and/or strategy adaptation schemes. These steps have been inspired by many adaptive algorithms already presented in literature. This constitutes also the various components of these algorithms, including selection, recombination, mutation, and survival operators.

Framework 1: General framework for constructing an adaptive metaheuristic with respect to parameters control and perturbation strategies.

Step 1: Individual (Solution) Encoding, Population Representation and Self-Adaptive Parameters

A population is a set (i.e. array) of individuals (chromosomes, particles) where *Np*, refers to the number of individuals in each generation. First, we have to encode the necessary information required for the problem analysis in the individual structure. Each individual should represent a complete solution to the problem at hand. All together, these parameters are called solution parameters [39]. Additionally, if our intention is to work on self-adaptive algorithm, the individual should implicitly include also the encoding of the control parameter(s), or the so-called strategy's parameter(s) that we would to undergo evolution via recombination and mutation. This requires an intelligent decision to be taken into account in such a way that better parameter values would tend to produce better individuals (i.e. solutions) with the highest chance to survive and propagate for more offsprings [18,19].

Step 2: Individual Evaluation (Solution Validation) and Selection

The definition of the fitness function is crucially important for a successful application. In any meta-algorithm, we have to evaluate the fitness of each individual. There are many standard fitness functions that can be used to test any proposed meta-method [27,124]. Throughout many problems, it is more natural to state the fitness function as minimization with/without constraints rather than maximization of some utility objectives; however this depends on the type of the problem/application at hand. This step is always connected to selection operation and survival of the fittest, as these two operations work dependent on the fitness evaluation [40,145].

Step 3: Individual Generator Strategy

One of the most important steps when using metaheuristic algorithms is how the candidate solutions will be changed in order to diversify the population with new solutions. Exploration and exploitation are the two cornerstones of problem solving by search. For a successful evolutionary process, one has to get proper balance between exploration (to cover sufficiently the solution space seeking out for better solutions), and exploitation (refining the solutions by combining information gathered from good ones during the exploration phase). Also, diversity maintenance is important to prevent premature convergence. This gives a motivation to study and provide sufficient exploration and exploitation techniques to be incorporated into the meta-algorithm main paradigm [85,89]. The main individual generation strategies that should get a thorough understanding and then well applied are: recombination and mutation. Several individual strategies can be involved in the same algorithm. The selection of the right individual generator can be implemented implicitly during the run. This requires additional approaches that collect information during evolution in order to make decision which is the best strategy to be applied for an epoch [53,139].

Step 4: Stopping Criteria

The most common stopping condition used in the literature is to allow the algorithm to run to a maximum number of iterations [39]. A small number of iterations may not offer enough time for the algorithm to attain an optimum especially when the size of the search space is large. On the other hand, once the optimum solution is reached there will be no more gain from a very large number of iterations. In general, the maximum number of iterations allowed to multiply by the swarm size gives an indication of the number of particles evaluated by the algorithm [21].

Step 5: Investigate the Influence of the Algorithm Parameters on the performance of the Developed Metaheuristic Algorithm

This includes:

- Population size *Np*. To draw the effect of *Np* on the population diversity and performance of the proposed meta-algorithm, also to see whether increasing the size of the population may prevent, or at least, reduce the chance of the algorithm to be trapped in local minima [11,125].
- Selection and perturbation operators (recombination and mutation) with their qualitative and quantitative parameter setting. A determination should be made in advance to the type of parameter control settings that would be changed in deterministic, adaptive and/or self-adaptive manner, the change evidence, and for which parameter(s) e.g. mutation factor, crossover rate, tournament size, and so on [23,135].
- Determine the scope of change which is either to be on the gene level, individual level, or the whole population level [18,31].

Step 6: Experimental Results

This step is performed to reach twofold goals. First, to show the reliability and efficiency of the proposed methods, experimental results will be evaluated using various test bed problems [27]. Second, the relative performance of the proposed methods, when compared with other well-known state-of-the-art algorithms should be evaluated (both in qualitative and/or quantitative terms), and reported when applied to the same test problem/application. In all these experiments, the influences of different versions, components, and parameters of the meta-algorithm on performance are to be investigated, analyzed, and reported [93,146].

7. Summary, conclusion and future insights

The classification of adaptive metaheuristics and adaptive differential evolution (DE) algorithms is always an on-going research area. The research on developing adaptive meta-algorithms has been such a hot topic. As more and more new adaptive algorithms are proposed with new characteristics, the need for a general classification that can cover all these types of algorithms becomes a large demand. These classifications provide knowledge to those researches who are interested in this type of algorithms on what have been implemented and what improvements or developments can be added in this area as future work. In this study, two classifications have been proposed for the purpose: First, extension taxonomy to the EA parameter settings that covers in general the type of parameters settings in evolutionary computations (Section 3). Second, general classification to the adaptive DE algorithms that classifies these algorithms based on the parameters control of the algorithm as well as the number of DE strategies employed in the implementation (Section 5). Also, this study has implemented a number of tables (Tables 2–8) after a thorough algorithmic design investigation applied on 28 recent and important adaptive DE variants. These tables summarize everything related to the algorithmic design of these algorithms, and the points of weaknesses and strengths of each algorithm.

In summary, the main focus of our literature analysis is the algorithmic design and we attempt to offer a protocol for future DE implementations (Section 6). In addition, many studies can be conducted to extend or enhance the adaptive DE algorithms we have presented, some of these future directions are stated in Table 9.

References*

[1] H.A. Abbass, The self-adaptive Pareto differential evolution algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC2002), Honolulu, HI, May 12-17, IEEE Press, 2002, pp. 831–836.

[2] H. Abderazek, D. Ferhat, A. Ivana, Adaptive mixed differential evolution algorithm for bi-objective tooth profile spur gear optimization, Int. J. Adv. Manuf. Technol. 90 (2017) 2063–2073.

[3] M.D. Al-Dabbagh, R.D. Al-Dabbagh, R.S.A. Raja Abdullah, F. Hashim, A new modified differential evolution algorithm scheme-based linear frequency modulation radar signal de-noising, Eng. Optim. 74 (2015) 771–787, https://doi.org/10.1080/0305215x.2014.927449.

[4] R.D. Al-Dabbagh, S. Mekhilef, M.S. Baba, Parameters' fine tuning of differential evolution algorithm, Comput. Syst. Sci. Eng. 30 (2015) 125–139.

[5] M. Ali, P. Siarry, M. Pant, An efficient Differential Evolution based algorithm for solving multi-objective optimization problems, Eur. J. Oper. Res. 217 (2012) 404–416.

[6] M.M. Ali, A. Törn, Population set based global optimization algorithms: some modifications and numerical studies, Comput. Oper. Res. 31 (2004) 1703–1725.

[7] P.J. Angeline, Adaptive and self-adaptive evolutionary computations, in: Computational Intelligence: a Dynamic Systems Perspective, 1995, pp. 152–163.

[8] D.V. Arnold, H.-G. Beyer, Noisy Optimization with Evolution Strategies. Genetic Algorithms and Evolutionary Computation, Springer Science & Business Media, Dortmund, Germany, 2002.

[9] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble of parameters in a sinusoidal differential evolution with niching-based population, Reduc. Swarm Evol. Comput. (2017), https://doi.org/10.1016/j.swevo.2017.09.009.

[10] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.

[11] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems, in: Paper Presented at the IEEE Congress on Evolutionary Computation (CEC 2016), Vancouver, BC, Canada, 24–29 July, 2016.

[12] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, A.M. Shatnawi, A novel differential crossover strategy based on covariance matrix learning with Euclidean neighborhood for solving real-world problems, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.

[13] S. Baluja, Population-based Incremental Learning: a Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Carnegie Mellon University, Pittsburgh, PA, 1994.

* Note: Refs. [30], [33] and [121] have been omitted because they are redundant references.

[14] S. Baluja, R. Caruana, Removing the Genetics from the Standard Genetic Algorithm, Carnegie Mellon University, Pittsburgh, PA, 1995.

[15] X.-J. Bi, J. Xiao, Classification-based self-adaptive differential evolution with fast and reliable convergence performance, Soft Comput. 15 (2011) 1581–1599, https://doi.org/10.1007/s00500-010-0689-5.

[16] E. Bor, M. Turduev, H. Kurt, Differential evolution algorithm based photonic structure design: numerical and experimental verification of subwavelength λ/5 focusing of light, Sci. Rep. 6 (2016).

[17] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82–117.

[18] J. Brest, B. Boskovic, S. Greiner, V. Zumer, M.S. Maucec, Performance comparison of self-adaptive and adaptive differential evolution algorithms, Soft Comput. 11 (2007) 617–629, https://doi.org/10.1007/s00500-006-0124-0.

[19] J. Brest, B. Boskovic, V. Zumer, An improved self-adaptive differential evolution algorithm in single objective constrained real-parameter optimization, in: Paper Presented at the 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain, 18–23 July, 2010.

[20] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657, https://doi.org/10.1109/tevc.2006.872133.

[21] J. Brest, M.S. Maucec, Self-adaptive differential evolution algorithm using population size reduction and three strategies, Soft Comput. 15 (2011) 2157–2174, https://doi.org/10.1007/s00500-010-0644-5.

[22] J. Brest, A. Zamuda, B. Boskovic, M.S. Maucec, V. Zumer, Dynamic optimization using self-adaptive differential evolution, in: Paper Presented at the IEEE Congress on Evolutionary Computation (CEC 2009), Trondheim, Norway, 18–21 May, 2009.

[23] P. Bujok, J. Tvrdík, Enhanced individual-dependent differential evolution with population size adaptation, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.

[24] P. Bujok, J. Tvrdík, R. Poláková, Differential evolution with rotation-invariant mutation and competing-strategies adaptation, in: Paper Presented at the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July, 2014.

[25] A. Caponio, F. Neri, Differential evolution with noise analyzer, in: Paper Presented at the Workshops on Applications of Evolutionary Computation (EvoWorkshops 2009), 2009.

[26] F. Caraffini, F. Neri, I. Poikolainen, Micro-differential evolution with extra moves along the axes, in: Proceedings of the 2013 Ieee Symposium on Differential Evolution (Sde), 2013.

[27] R. Cheng, M. Li, Y. Tian, X. Zhang, S. Yang, Y. Jin, X. Yao, Benchmark Functions for CEC'2017 Competition on Evolutionary Many-Objective Optimization, School of Computer Science, University of Birmingham, Birmingham, U.K, 2017.

[28] T.-C. Chiang, C.-N. Chen, Y.-C. Lin, Parameter control mechanisms in differential evolution, in: A Tutorial Review and Taxonomy Proceedings of the 2013 Ieee Symposium on Differential Evolution (Sde), 2013.

[29] C. Cotta, M. Sevaux, K.E. Sörensen, Adaptive and Multilevel Metaheuristics Vol. 136. Studies in Computational Intelligence, Springer-Verlag, Berlin,Germany, 2008.

[31] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (2009) 526–553.

[32] S. Das, A. Ghosh, S.S. Mullick, A switched parameter differential evolution for large scale global optimization – simpler may be better, in: Paper Presented at the Mendel 2015: 21st International Conference on Soft Computing, Advances in Intelligent Systems and Computing, Brno, Czech Republic, 23 – 25 June, 2015.

[34] S. Das, A. Mandal, R. Mukherjee, An adaptive differential evolution algorithm for global optimization in dynamic environments, IEEE Trans. Cybern. 44 (2014) 966–978.

[35] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution - an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.

[36] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (2011) 27–54, https://doi.org/10.1109/tevc.2010.2059031.

[37] S. Dominguez-Isidro, E. Mezura-Montes, The baldwin effect on a memetic differential evolution for constrained numerical optimization problems, in: Paper Presented at the Proceedings of the Genetic and Evolutionary Computation (GECOO 2017), Berlin, Germany 15–19 July, 2017.

[38] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (1999) 124–141, https://doi.org/10.1109/4235.771166.

[39] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing. Natural Computing Series, second ed., Springer-Verlag, Berlin,Germany, 2003.

[40] S. Elsayed, R. Sarker, C.C. Coello, T. Ray, Adaptation of operators and continuous control parameters in differential evolution for constrained optimization, Soft Comput. (2017), https://doi.org/10.1007/s00500-017-2712-6.

[41] S.M. Elsayed, R.A. Sarker, D.L. Essam, A self-adaptive combined strategies algorithm for constrained optimization using differential evolution, Appl. Math. Comput. 241 (2014) 267–282, https://doi.org/10.1016/j.amc.2014.05.018.

[42] H.Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, J. Global Optim. 27 (2003) 105–129, https://doi.org/10.1023/a:1024653025686.

[43] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with Zoning evolution of control parameters and adaptive mutation strategies, IEEE Trans. Cybern. 46 (2016) 219–232, https://doi.org/10.1109/TCYB.2015.2399478.

[44] L. Feng, W. Zhou, L. Zhou, J.H. Zhong, B.S. Da, Z.X. Zhu, Y. Wang, An empirical study of multifactorial PSO and multifactorial DE, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.

[45] V. Feoktistov, Differential evolution, in: Search of Solutions Vol. 5. Springer Optimization and its Applications, Springer-Verlag, New York, United State, 2006.

[46] V. Feoktistov, S. Janaqi, Generalization of the strategies in differential evolution, in: 18-th Annual IEEE International Parallel and Distributed Processing Symposium, Santa Fe, New Mexico, USA, April 26-30, IEEE Computer Society, 2004, pp. 2341–2346.

[47] V. Feoktistov, S. Janaqi, New strategies in differential evolution - design principle, in: I.C. Parmee (Ed.), Adaptive Computing in Design and Manufacture VI, Springer - Verlag London Limited, UK, London, 2004, pp. 335–346, https://doi.org/10.1007/978-0-85729-338-1_28.

[48] I. Fister, X.-S. Yang, I. Fister, J. Brest, D. Fister, A brief review of nature-inspired algorithms for optimization, doi:arXiv:1307.4186, Elektrotehniski Vestnik/Electrotech. Rev. 80 (2013) 1–7.

[49] D.B. Fogel, L.J. Fogel, J.W. Atmar, Meta-evolutionary programming, in: Conference on Signals, Systems and Computers. 1991 Conference Record of the Twenty-fifth Asilomar, San Diego, CA, USA, Nov 4-6, 1991, pp. 540–545, https://doi.org/10.1109/acssc.1991.186507.

[50] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm, Harmony Search Simul. 76 (2001) 60–68.

[51] M. Gendreau, J.-Y. Potvin, Handbook of Metaheuristics Vol. 146. International Series in Operations Research & Management Science, second ed., Springer, London, 2010.

[52] A. Ghosh, S. Das, A. Chowdhury, R. Giri, An improved differential evolution algorithm with fitness-based adaptation of the control parameters, Inf. Sci. 181 (2011) 3749–3765.

[53] A. Ghosh, S. Das, B.K. Panigrahi, A noise resilient differential evolution with improved parameter and strategy control, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.

[54] W. Gong, Z. Cai, C.X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Trans. Syst. Man Cybern. B Cybern. 41 (2011) 397–413, https://doi.org/10.1109/tsmcb.2010.2056367.

[55] S.K. Goudos, M. Deruyck, D. Plets, L. Martens, W. Joseph, Optimization of power consumption in 4G LTE Networks using a novel barebones self-adaptive differential evolution algorithm, Telecommun. Syst. 66 (2017) 109–120, https://doi.org/10.1007/s11235-017-0279-2.

[56] U. Halder, S. Das, D. Maity, A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments, IEEE Trans. Cybern. 43 (2013) 881–897.

[57] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20-22 May, 1996, pp. 312–317.

[58] S. Hui, P.N. Suganthan, Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization, IEEE Trans. Cybern. 46 (2016) 64–74.

[59] S.M. Islam, S. Das, S. Ghosh, S. Roy, 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. 42 (2012) 482–500, https://doi.org/10.1109/tsmcb.2011.2167966.

[60] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments-a survey, IEEE Trans. Evol. Comput. 9 (2005) 303–317.

[61] HAeO. Junior, L. Ingber, A. Petraglia, M.R. Petraglia, M.A.S. Machado, Global optimization and its applications, in: Stochastic Global Optimization and its Applications with Fuzzy Adaptive Simulated Annealing Vol. 35. Intelligent Systems Reference Library, Springer-Verlag, Berlin Heidelberg, 2012, pp. 11–20.

[62] HAeO. Junior, L. Ingber, A. Petraglia, M.R. Petraglia, M.A.S. Machado, Metaheuristic methods, in: Stochastic Global Optimization and its Applications with Fuzzy Adaptive Simulated Annealing, Vol. 35. Intelligent Systems Reference Library, Springer-Verlag, Berlin Heidelberg, 2012, pp. 21–30.

[63] G. Karafotias, M. Hoogendoorn, A.E. Eiben, Parameter control in evolutionary algorithms: trends and challenges, IEEE Trans. Evol. Comput. 19 (2015) 167–187.

[64] J. Kennedy, R. Eberhart, Particle swarm optimization, in: 1995 Proceedings of IEEE International Conference on Neural Networks, Perth, WA, Nov 27-Dec 01, IEEE Press, 1995, pp. 1942–1948.

[65] S. Kukkonen, E. Mezura-Montes, An experimental comparison of two constraint handling approaches used with differential evolution, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.

[66] J. Lampinen, Solving problems subject to multiple nonlinear constraints by the differential evolution, in: Proceedings of MENDEL'01-7th International Conference on Soft Computing, Brno, Czech Republic, June 2001, pp. 50–57.

[67] J. Lampinen, A constraint handling approach for the differential evolution algorithm, in: Proceedings of the 2002 Congress on Evolutionary Computation - CEC2002, Honolulu, HI, May 12-17, IEEE Press, 2002, pp. 1468–1473.

[68] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Proceedings of MENDEL'00-6th International Mendel Conference on Soft Computing, Brno, Czech Republic, 2000, pp. 76–83.

[69] J.L.J. Laredo, C. Fernandes, J.J. Merelo, C. Gagné, Improving genetic algorithms performance via deterministic population shrinkage, in: Paper Presented at the

11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09), Montreal, Québec, Canada, 08–12 July, 2009.
[70] M. Leon, N. Xiong, Adapting differential evolution algorithms for continuous optimization via greedy adjustment of control parameters, J. Artif. Intell. Soft Comput. Res. 6 (2016) 103–118.
[71] X. Li, S. Ma, J. Hu, Multi-search differential evolution algorithm, Appl. Intell. 47 (2017) 231–256.
[72] C. Lin, A. Qing, Q. Feng, A comparative study of crossover in differential evolution, J. Heuristics 17 (2011) 675–703, https://doi.org/10.1007/s10732-010-9151-1.
[73] J. Lis, Parallel genetic algorithm with dynamic control parameter, in: Proceedings of the 1996 IEEE Conference on Evolutionary Computation, Nagoya, Japan, IEEE Press, 1996, pp. 324–329.
[74] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Comput. 9 (2005) 448–462, https://doi.org/10.1007/s00500-004-0363-x.
[75] F.S. Lobato, V.S. Jr, Self-adaptive multi-objective optimization differential evolution, in: Multi-objective Optimization Problems. SpringerBriefs in Mathematics, Springer, Cham, 2017, pp. 47–73, https://doi.org/10.1007/978-3-319-58565-9_4.
[76] F.G. Lobo, C.F. Lima, Z. Michalewicz, Parameter Setting in Evolutionary Algorithms Vol. 54. Studies in Computational Intelligence, Springer-Verlag, Berlin, Germany, 2007.
[77] N. Lynn, R. Mallipeddi, P.N. Suganthan, Differential evolution with two Subpopulations, in: Paper Presented at the International Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO 2014), Bhubaneswar, India, 18–20 Dec, 2014.
[78] R. Mallipeddi, Harmony search based parameter ensemble adaptation for differential evolution, J. Appl. Math. (2013), https://doi.org/10.1155/2013/750819.
[79] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679–1696, https://doi.org/10.1016/j.asoc.2010.04.024.
[80] E. Mezura-Montes, M.E. Miranda-Varela, RdC. Gómez-Ramón, Differential evolution in constrained numerical optimization: an empirical study, Inf. Sci. 180 (2010) 4223–4262.
[81] E. Mezura-Montes, M. Reyes-Sierra, C.A.C. Coello, Multi-objective optimization using differential evolution: a survey of the state-of-the-art, in: Advances in Differential Evolution, Vol. 143. Studies in Computational Intelligence, Springer-Verlag Berlin Heidelberg, 2008, pp. 173–196.
[82] E. Mininno, F. Neri, F. Cupertino, D. Naso, Compact differential evolution, IEEE Trans. Evol. Comput. 15 (2011) 32–54, https://doi.org/10.1109/tevc.2010.2058120.
[83] A.W. Mohamed, An improved differential evolution algorithm with triangular mutation for global numerical optimization, Comput. Ind. Eng. 85 (2015) 359–375. https://doi.org/10.1016/j.cie.2015.04.012.
[84] A.W. Mohamed, A novel differential evolution algorithm for solving constrained engineering optimization problems, J. Intell. Manuf. (2017) 1–34, https://doi.org/10.1007/s10845-017-1294-6.
[85] A.W. Mohamed, Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm, Complex Intell. Syst. 3 (2017) 205–231, https://doi.org/10.1007/s40747-017-0041-0.
[86] A.W. Mohamed, A.A. Hadi, A.M. Fattouh, K.M. Jambi, LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.
[87] A.W. Mohamed, A.K. Mohamed, Adaptive guided differential evolution algorithm with novel mutation for numerical optimization, Int. J. Mach. Learn. Cybern. (2017) 1–25, https://doi.org/10.1007/s13042-017-0711-7.
[88] A.W. Mohamed, H.Z. Sabry, Constrained optimization based on modified differential evolution algorithm, Inf. Sci. 194 (2012) 171–208.
[89] A.W. Mohamed, P.N. Suganthan, Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation, Soft Comput. (2017) 1–21, https://doi.org/10.1007/s00500-017-2777-2.
[90] O.G. Monakhov, E.A. Monakhova, M. Pant, Application of differential evolution algorithm for optimization of strategies based on financial time series, Numer. Anal. Appl. 9 (2016) 150–158.
[91] E.M. Montes, C.A. Coello Coello, E.I. Tun-Morales, Simple feasibility rules and differential evolution for constrained optimization, in: Proceedings of the Third Mexican International Conference on Artificial Intelligence (MICAI'2004), New York, Springer-Verlag, April 2004, pp. 707–716.
[92] F. Neri, E. Mininno, Memetic compact differential evolution for cartesian robot control, IEEE Comput. Intell. Mag. 5 (2010) 54–65, https://doi.org/10.1109/mci.2010.936305.
[93] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (2010) 61–106, https://doi.org/10.1007/s10462-009-9137-2.
[94] F. Neri, V. Tirronen, On memetic differential evolution frameworks: a study of advantages and limitations in hybridization, in: 2008 Ieee Congress on Evolutionary Computation, vols. 1–8, Ieee, 2008, pp. 2135–2142, https://doi.org/10.1109/cec.2008.4631082.
[95] P. Novoa-Hernández, C.C. Corona, D.A. Pelta, Self-adaptive, multipopulation differential evolution in dynamic environments, Soft Comput. - A Fusion Found. Methodol. Appl. 17 (2013) 1861–1881.
[96] B.K. Panigrahi, Y. Shi, M.-H. Lim, Handbook of Swarm Intelligence: Concepts, Principles and Applications Vol. 8. Adaptation, Learning, and Optimization, Springer, Chennai, India, 2011.
[97] F. Peng, K. Tang, G. Chen, X. Yao, Multi-start JADE with knowledge transfer for numerical optimization, in: 2009 Ieee Congress on Evolutionary Computation (CEC '09), Trondheim, Norway, May 18-21, 2009, pp. 1889–1895.
[98] R. Poláková, L-SHADE with competing strategies applied to constrained optimization, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.
[99] R. Polakova, J. Tvrdık, P. Bujok, Population-size adaptation through diversity-control mechanism for differential evolution, in: Paper Presented at the MENDEL, 22th International Conference on Soft Computing, Brno, Czech Republic, 2016.
[100] K. Price, R. Storn, Differential evolution: a simple evolution strategy for fast optimization Dr Dobb's, J. Softw. Tools 22 (1997) 18–24.
[101] K.V. Price, Differential evolution vs. the functions of the 2nd ICEO, in: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97), Indianapolis, IEEE; IEEE Neural Network Council (NNC), April 1997, pp. 153–157.
[102] K.V. Price, An Introduction to differential evolution, in: New Ideas in Optimization, McGraw-Hill, London,, 1999.
[103] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: a Practical Approach to Global Optimization. Natural Computing Series, first ed., Springer-Verlag, Berlin, Germany, 2005.
[104] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417, https://doi.org/10.1109/tevc.2008.927706.
[105] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of the 2005 Ieee Congress on Evolutionary Computation, Edinburgh, SCOTLAND, SEP 02-05, IEEE; IEEE Computat Intelligence Soc; IEE; Evolut Programming Soc, 2005, pp. 1785–1791.
[106] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution algorithms, in: 2006 IEEE Congress on Evolutionary Computation, Vancouver, Canada, 16-21, IEEE Press, July 2006, pp. 1995–2002.
[107] P. Rakshit, A. Chowdhury, A. Konar, Differential evolution induced many objective optimization, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.
[108] I. Rechenberg, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, 1973.
[109] T. Robič, B. Filipič, DEMO: differential evolution for multiobjective optimization, in: Paper Presented at the International Conference on Evolutionary Multi-criterion Optimization (EMO 2005), 2005.
[110] J. Rönkkönen, S. Kukkonen, V.K. Price, Real-parameter optimization with differential evolution, in: The 2005 IEEE Congress on Evolutionary Computation CEC 2005, IEEE press, Edinburgh, Scotland, Sep 5 2005, pp. 506–513.
[111] N. Salvatore, A. Caponio, F. Neri, S. Stasi, G.L. Cascella, Optimization of delayed-state Kalman-Filter-based algorithm via differential evolution for sensorless control of induction motors, IEEE Trans. Ind. Electron. 57 (2010) 385–394.
[112] R.A. Sarker, S.M. Elsayed, T. Ray, Differential evolution with dynamic parameters selection for optimization problems, IEEE Trans. Evol. Comput. 18 (2014) 689–707.
[113] H.-P. Schwefel, Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie, Birkhäuser, 1977.
[114] K. Sindhya, S. Ruuska, T. Haanpaa, K. Miettinen, A new hybrid mutation operator for multiobjective optimization with differential evolution, Soft Comput. 15 (2011) 2041–2055, https://doi.org/10.1007/s00500-011-0704-5.
[115] R.E. Smith, E. Smuda, Adaptively resizing populations: algorithm, analysis and first results, Complex Syst. 9 (1995) 47–72.
[116] R. Storn, On the usage of differential evolution for function optimization, in: Biennial Conference of the North America Fuzzy Information Processing Society (NAFIPS 1996), IEEE, Berkeley, CA, New York, 1996, pp. 519–523.
[117] R. Storn, Real-world applications in the communications industry - when do we resort to Differential Evolution?, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.
[118] R. Storn, K. Price, Differential Evolution: a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, International Computer Science Institute (ICSI), USA, 1995.
[119] R. Storn, K. Price, Minimizing the real functions of the ICEC'96 contest by differential evolution, in: IEEE International Conference on Evolutionary Computation, May 1996, IEEE, Nagoya, New York, 1996, pp. 842–844.
[120] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[122] P.N. Suganthan, Future of real parameter optimization, in: Paper Presented at the International Conference on Soft Computing for Problem Solving (SocProS 2016) Thapar University, Topiala, India, 23–24 Dec, 2016.
[123] P.N. Suganthan, S. Das, S. Mukherjee, S. Chatterjee, Adaptation methods in differential evolution: a review, in: Paper Presented at the 20th International Conference on Soft Computing MENDEL 2014, Brno, Czech Republic, 25–27 June, 2014.
[124] R. Tanabe, A. Fukunaga, Evaluating the performance of SHADE on CEC 2013 benchmark problems, in: Paper Presented at the IEEE Congress on Evolutionary Computation (CEC 2013), Cancun, Mexico, 20–23 June, 2013.
[125] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population reduction, in: Paper Presented at the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July, 2014.
[126] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, Soft Comput. 10 (2006) 673–686, https://doi.org/10.1007/s00500-005-0537-1.

[127] L.-Y. Tseng, C. Chen, Multiple trajectory search for multiobjective optimization, in: 2007 Ieee Congress on Evolutionary Computation, Singapore, Singapore, Sep 25-28, 2007, pp. 3609–3616.
[128] L.-Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, in: 2008 Ieee Congress on Evolutionary Computation, Hong Kong, Peoples R China, Jun 01-06, 2008, pp. 3052–3059.
[129] J. Tvrdik, Adaptation in differential evolution: a numerical comparison, Appl. Soft Comput. 9 (2009) 1149–1155, https://doi.org/10.1016/j.asoc.2009.02.010.
[130] V. Uher, P. Gajdoš, M. Radecký, V. Snášel, Utilization of the discrete differential evolution for optimization in multidimensional point clouds, Comput. Intell. Neurosci. 2016 (2016) 1–14.
[131] O. Urfalioglu, O. Arikan, Self-adaptive randomized and rank-based differential evolution for multimodal problems, J. Global Optim. 51 (2011) 607–640, https://doi.org/10.1007/s10898-011-9646-9.
[132] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, Archive analysis in SHADE, in: Paper Presented at the International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 2017.
[133] H. Wang, S. Rahnamayan, Z. Wu, Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems, J. Parallel Distr. Comput. 73 (2013) 62–73, https://doi.org/10.1016/j.jpdc.2012.02.019.
[134] X. Wang, Z. Dong, L. Tang, A multi-objective differential evolution algorithm with memory based population construction, in: Paper Presented at the 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, Canada, 24–29 July, 2016.
[135] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66, https://doi.org/10.1109/tevc.2010.2087271.
[136] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inf. Sci. 185 (2012) 153–177, https://doi.org/10.1016/j.ins.2011.09.001.
[137] M. Weber, F. Neri, V. Tirronen, Shuffle or update parallel differential evolution for large-scale optimization, Soft Comput. 15 (2011) 2089–2107, https://doi.org/10.1007/s00500-010-0640-9.
[138] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Comput. 14 (2010) 1187–1207, https://doi.org/10.1007/s00500-009-0510-5.
[139] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P.N. Suganthan, Ensemble of differential evolution variants, Inf. Sci. 423 (2018) 172–186.
[140] X.-S. Yang, Z. Cui, R. Xiao, A.H. Gandom, M. Karamanoglu, Swarm Intelligence and Bio-inspired Computation: Theory and Applications, Elsevier,, 2013.
[141] X. Yu, W.-N. Chen, X.-M. Hu, J. Zhang, Fast 3D Path Planning based on heuristic-aided differential evolution, in: Paper Presented at the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2017), Berlin, July 15–19, 2017.
[142] D. Zaharie, Parameter adaptation in differential evolution by controlling the population diversity, in: Proc. Of SYNASC'2002, Analele Univ. Timisoara, Timisoara, Roumania, 2002, pp. 281–295 seria Matematica-Informatica, vol special issue.
[143] D. Zaharie, A multi-population differential evolution algorithm for multi-modal optimization, in: Mendel'04-10th International Conference on Soft Computing, Brno, Czech Republic, June 2004, pp. 17–22.
[144] D. Zaharie, A comparative analysis of crossover variants in differential evolution, in: Proceedings of the IMCSIT, 2nd International Symposium Advances in Artificial Intelligence and Applications (AAIA'07), 2007, pp. 171–181.
[145] A. Zamuda, Adaptive constraint handling and success history differential evolution for CEC 2017 constrained real-parameter optimization, in: Paper Presented at the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June, 2017.
[146] A. Zamuda, J. Brest, Self-adaptive control parameters' randomization frequency and propagations in differential evolution, Swarm Evol. Comput. 25 (2015) 72–99.
[147] G. Zhang, L. Pan, F. Neri, M. Gong, A. Leporati, Metaheuristic optimization: algorithmic design and applications, J. Optim. 2017 (2017) 1–2, https://doi.org/10.1155/2017/1053145.
[148] J. Zhang, A.C. Sanderson, Adaptive Differential Evolution: a Robust Approach to Multimodal Problem Optimization Vol. 1. Adaptive, Learning, and Optimization, Springer-Verlag, Chennai, India, 2009.
[149] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958, https://doi.org/10.1109/tevc.2009.2014613.
[150] X. Zhang, W. Chen, C. Dai, W. Cai, Dynamic multi-group self-adaptive differential evolution algorithm for reactive power optimization, Int. J. Electr. Power Energy Syst. 32 (2010) 351–357, https://doi.org/10.1016/j.ijepes.2009.11.009.
[151] S.-Z. Zhao, P.N. Suganthan, S. Das, Self-adaptive differential evolution with multi-trajectory search for large-scale optimization, Soft Comput. 15 (2011) 2175–2185, https://doi.org/10.1007/s00500-010-0645-4.
[152] S. Zhao, X. Wang, L. Chen, W. Zhu, A novel self-adaptive differential evolution algorithm with population size adjustment scheme, Arabian J. Sci. Eng. 39 (2014) 6149–6174, https://doi.org/10.1007/s13369-014-1248-7.
[153] W. Zhu, Y. Tang, J-a Fang, W. Zhang, Adaptive population tuning scheme for differential evolution, Inf. Sci. 223 (2013) 164–191, https://doi.org/10.1016/j.ins.2012.09.019.
[154] D. Zou, H. Liu, L. Gao, S. Li, A novel modified differential evolution algorithm for constrained optimization problems, Comput. Math. Appl. 61 (2011) 1608–1623, https://doi.org/10.1016/j.camwa.2011.01.029.
[155] A. Zamuda, J. Brest, E. Mezura-Montes, Structured Population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization, in: Paper Presented at the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20-23 June, 2013.
[156] A. Zamuda, J. Brest, Population reduction differential evolution with multiple mutation strategies in real world industry challenges, in: Paper presented at the Swarm and Evolutionary Computation-International Symposia, SIDE 2012 and EC 2012, Zakopane, Poland, 2012.