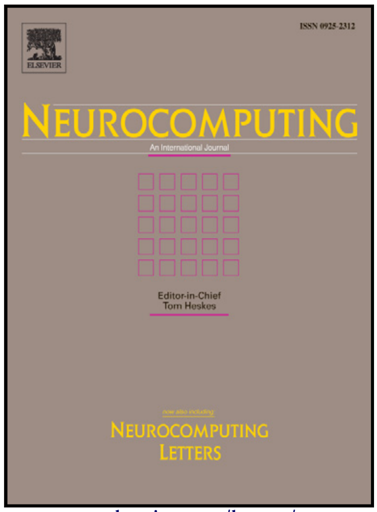
A new self-adaptation scheme for differential evolution

Lu, Xiaofen; Tang, Ke; Sendhoff, Bernhard; Yao, Xin

DOI:
10.1016/j.neucom.2014.04.071



www.elsevier.com/locate/neucom

A new self-adaptation scheme for differential evolution

Xiaofen Lu, Ke Tang, Bernhard Sendhoff, Xin Yao

PII: S0925-2312(14)00882-0
DOI: http://dx.doi.org/10.1016/j.neucom.2014.04.071
Reference: NEUCOM14423

To appear in: *Neurocomputing*

Received date: 15 November 2013
Revised date: 7 March 2014
Accepted date: 3 April 2014

A New Self-adaptation Scheme for Differential Evolution

Xiaofen Lu[a,c,*], Ke Tang[a,*], Bernhard Sendhoff[b], Xin Yao[a,c]

[a] *USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications School of Computer Science and Technology University of Science and Technology of China (USTC), Hefei, Anhui 230027, China.*
[b] *Honda Research Institute Europe GmbH, Offenbach 63073, Germany.*
[c] *Center of Excellence for Research in Computational Intelligence and Applications (CERCIA) School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K.*

Abstract

The performance of Differential Evolution (DE) largely depends on the choice of trial vector generation strategy and the values of its control parameters. In the past years, quite a few DE variants have been developed to adaptively adjust the strategy and control parameters during the search process. However, these variants may not perform satisfactorily when coping with computationally expensive problems (CEPs), for which a satisfying solution needs to be obtained with very limited fitness evaluations (FEs). In this paper, we demonstrate that not only can surrogate models be used to approximate the fitness function, they can also provide a good alternative method to adapt the strategy and control parameters of DE, and thus propose a framework called DE with Surrogate-assisted Self-Adaptation (DESSA). DESSA generates multiple trial vectors using different trial vector generation strategies and parameter settings, and then employs a surrogate model to identify the potentially best trial vector to undergo real fitness evaluation. As each trial vector corresponds to a unique combination of strategy and parameter setting, the surrogate model acts like a strategy/parameter setting selector that aims to identify the most suitable strategy and parameter setting for each target vector. Since DESSA can be easily combined with different DE variants, three concrete DE variants, namely DESSA-CoDE, DESSA-SaDE, and DESSA-CoDE*, are proposed. Comprehensive empirical studies demonstrate that DESSA can lead to superior performance over the compared adaptive DE variants. More important, it is shown that DESSA has the potential of accommodating more search strategies, which may lead to novel DE variants with even more competitive performance.

Keywords: Differential Evolution, Self-adaptation, Computationally Expensive Problems, Surrogate Model.

*Corresponding author
Email addresses: xiaofen@mail.ustc.edu.cn (Xiaofen Lu), ketang@ustc.edu.cn (Ke Tang), bernhard.sendhoff@honda-ri.de (Bernhard Sendhoff), x.yao@cs.bham.ac.uk (Xin Yao)

1. Introduction

Differential Evolution (DE), proposed by Storn and Price in 1995 [1], is well recognized as an Evolutionary Algorithm (EA) for solving real-parameter optimization problems. Owing to its simplicity and powerful search ability, DE has got a wide variety of real-world applications and exhibited excellent performance on many problems in diverse fields [2, 3, 4].

For DE, there exist many trial vector generation strategies and different problems may prefer different strategies [5]. Moreover, the control parameter settings also have great influence on DE's performance [6]. Therefore, when using DE to solve a particular problem, it is generally necessary to try through various strategies and fine-tune the control parameters. However, such a trial-and-error procedure is often very time-consuming. To address this issue, researcher have proposed numerous DE variants in the past years [7, 8, 9, 10, 5, 11, 4, 12, 13, 14].

The majority of these DE variants intended to design self-adaptation schemes that can automatically find the suitable strategy or parameter setting during the search process, such as jDE [9], self-adaptive differential evolution (SaDE) [8, 5], self-adaptive differential evolution with neighborhood search (SaNSDE) [10], JADE [11], PM-AdapSS-DE [15], generalized adaptive DE (GaDE) [12], DE with Fitness-based Area-Under-Curve Bandit (F-AUC-Bandit) [16], and ensemble of mutation strategies and control parameters in DE (EPSDE) [13]. Though they look different from one another, the self-adaptation schemes of these DE variants can be considered designed with a similar methodology. That is, the strategy or control parameters are adjusted according to the information accumulated during the search process, and those with better performance in the previous generations are more likely to be used for generating new trial vectors.

More recently, a novel DE variant, namely composite DE (CoDE), was proposed in [14]. The underlying concept of CoDE is very different from the above-mentioned DE variants. To be specific, CoDE does not make use of self-adaptation schemes but relies on researchers' experience [14]. It adopts three well-studied strategies and three control parameter settings. For each individual (called target vector) in the current population, CoDE generates one trial vector using each strategy with a randomly selected parameter setting. Then, the three generated trial vectors are evaluated with the fitness function and the best one is reserved as the final trial vector. The empirical studies in [14] showed that CoDE outperformed several well-known DE variants and non-DE variants.

Although the above-mentioned efforts have significantly advanced the potential of DE, those DE variants do not meet the requirements of computationally expensive problems (CEPs), which broadly exist in complex engineering design fields [17, 18, 19, 20, 21]. In CEPs, one fitness evaluation can take many hours of computer time. For example, one function evaluation involving the solution of the Navier-Stokes equations can take many hours of computer time in aerodynamic wing design [20]. Therefore, for such problems, a satisfactory solution is usually required to be obtained using very limited number of fitness evaluations (FEs). However, the aforementioned self-adaptation schemes may require a lot of FEs to accumulate sufficient information for reliable self-adaptation. The simulation results on low dimensional problems in [11] have indicated that the adaptation scheme of JADE did not function efficiently within the small number of generations. Moreover, parameters or strategy in SaDE and SaNSDE are only adapted after some learning period, e.g. 50 generations in [5]. On the other hand, CoDE requires multiple (i.e., three) FEs to generate an offspring each time, and thus is not cost-effective in the context of CEPs. Actually, it would be better if it can be found out what makes a best strategy/parameter setting for different problems like the innovation method [22]. The innovation method was proposed to unveil salient knowledge about properties which make a solution optimal by analyzing the commonality and difference of a set of near-Pareto-optimal solutions for the multi-objective optimization problem. The information can later be employed to solve a new related optimization problem at hand. However, it is much more complicated to analyze why a trial vector generation strategy/control parameter setting is best, which depends on not only the fitness landscape but also the evolutionary state.

In this paper, we propose that surrogate models can provide a good method of adapting the trial vector generation strategy and control parameters of DE, and a framework named DE with Surrogate-assisted Self-Adaptation (DESSA) is proposed. Surrogate models are computationally efficient models, and can be used in lieu of the real fitness function to reduce computational cost [20, 23]. For example, surrogate models can be interpolation or regression models that are built to approximate the real fitness function using some input output pairs evaluated by the fitness function. The main idea of DESSA is to maintain a pool of trial vector generation strategies and a pool of control parameter settings. For each target vector in the current population, a trial vector is generated using each combination of strategy and parameter setting. Then, a surrogate model is built and used to pick out the most promising trial vector, which will be regarded as the final trial vector and undergo real fitness evaluation. It is important to note that the use of surrogate models in this paper is very different from the sole purpose of fitness approximation, which was done in a lot of existing work, and the key role of surrogate model in our paper is to select the most promising combination of trial vector generation strategy and control parameter setting. The motivation here actually shares some similarity to those behind IFEP [24], where it tried to identify the most promising mutation operator, and PAP [25], where it tried to identify the most promising algorithm.

Similar to the other self-adaptive variants of DE, DESSA also makes use of the information accumulated during the search process. However, instead of adapting the strategy and parameter settings based on their performance in the previous generations, DESSA directly focuses on employing surrogate models to compare different trial vectors that are generated using different strategies and parameter settings. It is well acknowledged that the performance of a strategy/parameter setting may change during the search process [13]. Therefore, predicting the performance of strategies/parameter settings can be expected to be more difficult than modeling the real fitness function. In other words, the latter task, though still non-trivial, might require fewer FEs to obtain a model that is beneficial and thus can suit the CEPs better. When compared to CoDE, DESSA only requires one FE in generating an offspring, and hence can be more cost-effective.

In fact, some initial studies have been conducted to incorporate surrogate models into DE in the literature [26, 27, 28, 29, 30, 31]. However, none of them investigate the utility of surrogate model from the perspective of self-adaptation. Moreover, most of these studies were dedicated to specific DE variants. In contrast, this paper mainly concerns the role of surrogate models from the perspective of self-adaptation, and its contributions include:

- It is suggested that a surrogate model might provide a promising way for the self-adaptation of the trial vector generation strategy and control parameters of DE.
- A new self-adaptation scheme that employs surrogate model is proposed, which is conceptually different from existing self-adaptation schemes for DE.
- The potential of the proposed self-adaptation scheme is explored by combining it with different DE variants, which clearly demonstrates that this new scheme can be combined with any DE variant that involves multiple search strategies.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to DE and some state-of-the-art DE variants. In Section 3, the DESSA framework and its instantiation based on CoDE are described. In Section 4, experimental results and analysis are presented to evaluate the efficacy of the DESSA framework. Finally, Section 5 concludes this paper.

2. Related Work

In this section, the framework of DE, several representatives of self-adaptive DE algorithms, and CoDE will be briefly reviewed. Interested readers are referred to [4] for a comprehensive survey on recent advance in DE.

2.1. *Differential Evolution (DE) Algorithm*

Without loss of generality, we assume the optimization problem has the following formulation.

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x}$ is a vector of n design variables in a continuous decision space $\Omega = \prod_{i=1}^{n} [L_i, U_i]$, and $f : \Omega \subseteq \Re^n \rightarrow \Re$ is called the objective function.

The procedure of DE for solving such optimization problems is given in Algorithm 1. DE begins with a randomly generated population in the decision space, $\boldsymbol{P}_G = \{\boldsymbol{x}_{i,G} | i = 1, 2, ..., popsize\}$. Then, DE iteratively uses the trial vector generation strategy (i.e., mutation and crossover operators) and the selection operator to evolve the population until a stopping criterion is met.

For the mutation operator, there are five frequently used mutation schemes for generating a mutant vector:

- "DE/rand/1" [2]

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r_1,G} + F \cdot (\boldsymbol{x}_{r_2,G} - \boldsymbol{x}_{r_3,G}) \tag{2}$$

- "DE/best/1" [2]

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{best,G} + F \cdot (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}) \tag{3}$$

Algorithm 1 The Framework of DE

Initialize a population $\boldsymbol{P}_G = \{\boldsymbol{x}_{i,G} | i = 1, 2, ..., popsize\}$
Evaluate $\boldsymbol{P}_G$
while the stopping criterion is not met **do**
  for each $\boldsymbol{x}_{i,G}$ in $\boldsymbol{P}_G$ **do**
    $\boldsymbol{v}_{i,G}$ = Mutate($\boldsymbol{P}_G$)
    $\boldsymbol{u}_{i,G}$ = Crossover($\boldsymbol{x}_{i,G}, \boldsymbol{v}_{i,G}$)
    $\boldsymbol{P}_{G+1} = \boldsymbol{P}_{G+1} \bigcup$ Select($\boldsymbol{x}_{i,G}$, $\boldsymbol{u}_{i,G}$)
  end for
  Set $G = G + 1$
end while

- "DE/target-to-best/1" [3]
  (i.e., "DE/rand-to-best/1" [2] or "DE/current-to-best/1" [11])

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + F \cdot (\boldsymbol{x}_{best,G} - \boldsymbol{x}_{i,G}) + F \cdot (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}) \quad (4)$$

- "DE/best/2 [2]

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{best,G} + F \cdot (\boldsymbol{x}_{r_1,G} - \boldsymbol{x}_{r_2,G}) + F \cdot (\boldsymbol{x}_{r_3,G} - \boldsymbol{x}_{r_4,G}) \quad (5)$$

- "DE/rand/2" [5]

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{r_1,G} + F \cdot (\boldsymbol{x}_{r_2,G} - \boldsymbol{x}_{r_3,G}) + F \cdot (\boldsymbol{x}_{r_4,G} - \boldsymbol{x}_{r_5,G}) \quad (6)$$

where the indices r_1, r_2, r_3, r_4, and r_5 are distinct integers randomly chosen from the range $[1, popsize]$ and also differ from i, and $\boldsymbol{x}_{best,G}$ is the best individual at the G-th generation. The parameter F is called the scale factor and typically ranges on the interval $[0.4, 1.0]$ according to [4].

For the crossover operator, there exist two crossover schemes for creating a trial vector with the mutant vector $\boldsymbol{v}_{i,G}$ and the target vector $\boldsymbol{x}_{i,G}$, i.e., exponential and binomial crossover schemes, and the latter is the more frequently used one, which can be described by the following formula:

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if } rand_j(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{j,i,G}, & \text{otherwise} \end{cases} \quad (7)$$

where $j = 1, 2, ..., n$, j_{rand} is a randomly selected integer $\in [1, n]$, $rand_j(0, 1)$ represents a number drawn uniformly between 0 and 1, $x_{j,i,G}$, $u_{j,i,G}$, and $v_{j,i,G}$ denote the j-th element of $\boldsymbol{x}_{i,G}$, $\boldsymbol{u}_{i,G}$, and $\boldsymbol{v}_{i,G}$, respectively. $CR \in [0, 1]$ is called the crossover rate.

In conjunction with the binomial crossover scheme, the above-mentioned mutation schemes yield a total of five trial vector generation strategies. They are "DE/rand/1/bin", "DE/best/1/bin", "DE/target-to-best/1/bin", "DE/best/2/bin", "DE/rand/2/bin", respectively.

After crossover, each generated trial vector $\boldsymbol{u}_{i,G}$ undergoes boundary constraint check. If the j-th element of $\boldsymbol{u}_{i,G}$ is out of the boundary, it is reset as follows:

$$u_{j,i,G} = \begin{cases} \min\{U_j, 2L_j - u_{j,i,G}\}, & \text{if } u_{j,i,G} < L_j \\ \max\{L_j, 2U_j - u_{j,i,G}\}, & \text{if } u_{j,i,G} > U_j \end{cases} \tag{8}$$

At last, the selection operator is performed to select the better one between $\boldsymbol{x}_{i,G}$ and $\boldsymbol{u}_{i,G}$ to enter the next generation:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} & \text{if } f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

2.2. DE with Self-Adaptation Schemes

As mentioned above, DE has multiple trial vector generation strategies and three control parameters, i.e., population size $popsize$, scale factor F, and crossover rate CR. Recognizing that some problems are very sensitive to the setting of them, researchers have investigated various self-adaptation schemes to automatically find the suitable settings during the search process. Usually, self-adaptation is applied to the trial vector generation strategy, F, and CR. In the following part of this subsection, the key points of some self-adaptive DE variants, jDE, JADE, SaDE, SaNSDE, PM-AdapSS-DE, DE with F-AUC-Bandit, GaDE, and EPSDE, are summarized in the following part of this subsection.

Brest *et al*. [9] proposed jDE for adapting F and CR, in which F and CR are encoded with the individual. It is believed that better control parameter values lead to better individuals that in turn, are more likely to survive. Thus, jDE adapts the encoded control parameters by propagating better parameter values to the next generation. Specifically, 90% of offspring individuals inherit the F or CR value from their separate parents at each generation. In other words, a successful F or CR value has the probability of 0.9 to be selected to generated an offspring at the next generation. Here, a successful F or CR value means that the offspring generated with this F or CR value successfully enters next the generation.

JADE [11] generates new F values according to a truncated Cauchy distribution and new CR values according to a normal distribution. The parameters of the Cauchy distribution and the normal distribution are updated using new successful F and CR values at each generation, respectively.

SaDE [8, 5], for the first time, adapts the trial vector generation strategy along with the control parameters. In SaDE, multiple trial vector generation strategies exist. The probability of selection of each strategy is updated proportionally to its *success rate* with relation to the others. The success rate of one strategy is the rate of the trial vectors generated by this strategy and successfully entering the next generation over previous learning period generations. Besides, SaDE generates a new CR for each target vector according to a Gaussian distribution, whose mean is updated every generation based on the successful CR values in previous learning period generations.

SaNSDE proposed by Yang *et al.* [10] can be considered as an improved version of SaDE. In SaNSDE, two different distributions are used to generate new F values. The probability of selection of each distribution is updated proportionally to its success rate with relation to the other. Also, the fitness improvement (the improvement achieved by the offspring over its parent) related to each successful CR is taken into consideration in SaNSDE in updating the mean of the Gaussian distribution for generating new CR.

PM-AdapSS-DE [15] keeps an empirical quality estimate for each trial vector generation strategy, and updates it based on the absolute average of the relative fitness improvement (normalized by the fitness of the best-so-far individual) recently received by the corresponding strategy. Then, the probability to select each strategy is updated proportionally to its empirical estimate with relation to the others.

In [16], the authors coupled a comparison-based technique, the F-AUC-Bandit [32], with DE to make DE adapt the strategy automatically and be invariant with relation to monotonous transformations over the fitness function. The generated DE variant keeps an empirical quality estimate for each strategy, which is updated by using the AUC paradigm and the rank of fitness of the offspring recently generated by the corresponding strategy. And, the probability to select each strategy is updated based on its empirical quality estimate.

In [12], a generalized parameter adaptation scheme was employed to design a new adaptive DE variant, GaDE, for large-scale optimization problems. In GaDE, new F and CR are generated for each target vector according to different probability distributions, whose parameters are updated every generation based on good F and CR values and the corresponding fitness improvements in previous generations, respectively.

Considering that different mutation strategies with different parameter settings can be appropriate during different stages of the evolution, Mallepeddi *et al.* [13] proposed EPSDE for adapting combination of mutation scheme and parameter setting. In EPSDE, combinations of mutation scheme and parameter setting are generated at the individual level and better combinations are propagated to the next generation.

2.3. CoDE

CoDE was proposed by Wang *et al.* [14] to improve DE through combining three trial vector generation strategies with three different control parameter settings, which have distinct advantages confirmed by other researchers' studies.

Specifically, the three strategies used in CoDE are:

- "DE/rand/1/bin"
- "DE/rand/2/bin"
- "DE/current-to-rand/1" [33],

and the three parameter settings are:

- $[F = 1.0, CR = 0.1]$
- $[F = 1.0, CR = 0.9]$
- $[F = 0.8, CR = 0.2]$.

In CoDE, for each target vector, a trial vector is generated using each strategy and a randomly selected parameter setting. Then, the three trial vectors will be evaluated with the real fitness function, and the best one is returned as the final trial vector.

In this section, we have introduced some representatives of DE variants that adjust the trial vector strategy and its control parameters during the search process. Besides these representatives, there are many other DE variants proposed in various research aspects. The interested readers are referred to [4] for details.

3. A new self-adaptation scheme using surrogate models

In this section, we propose a new scheme to adapt the strategy and control parameter setting for DE. The main idea of this scheme is to select the potentially best strategy and parameter setting by building surrogate models to select the most promising one among multiple trial vectors generated using several trial vector generation strategies and control parameter settings. Furthermore, a generalized framework of DE that employs the newly proposed scheme, DESSA, is proposed.

3.1. The DESSA Framework

Algorithm 2 DESSA

Initialize a population $\boldsymbol{P}_G = \{\boldsymbol{x}_{i,G} | i = 1, 2, ..., popsize\}$
Evaluate $\boldsymbol{P}_G$
Archive all exact evaluations into a database DB
while computational budget is not exhausted **do**
    if database building phase does not end **then**
        Evolve $\boldsymbol{P}_G$ with DE operators using exact evaluations
    else
        for each $\boldsymbol{x}_{i,G}$ in $\boldsymbol{P}_G$ **do**
            Generate multiple trial vectors using several strategies and parameter settings
            Build a surrogate model S based on DB
            Select the best trial vector (denoted as $\boldsymbol{u}_{i,G}$) according to S and evaluate it
            $\boldsymbol{P}_{G+1} = \boldsymbol{P}_{G+1} \bigcup$ Select($\boldsymbol{x}_{i,G}$, $\boldsymbol{u}_{i,G}$)
        end for
    end if
    Archive all exact evaluations into DB
    Set $G = G + 1$
end while

The outline of the proposed DESSA framework is given in Algorithm 2. DESSA begins with an initialized population of decision vectors. During the database building phase, the population is evolved with mutation, crossover and selection operators using exact evaluations for a certain number of generations, and all exact evaluations are archived into a database DB. After this, surrogate models are involved. For each target vector, several trial vector generation strategies and several control parameter settings are used to generate multiple trial vectors. Then, a surrogate model is built based on the evaluated points in the database. According to the surrogate model, the trial vector that appears the best is selected to undergo exact evaluation and competes with the target vector. This process is repeated until the computational budget is used up.

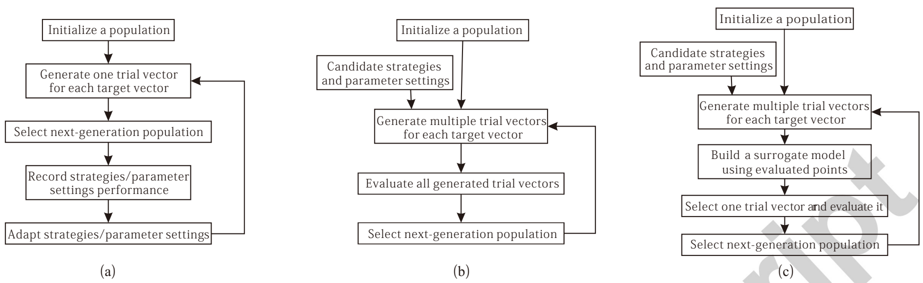


Figure 1: (a) A generalized framework for existing self-adaptive DE variants. (b) The generalized framework of CoDE. (c) The workflow of DESSA.

The main stages of DESSA are also illustrated in Fig. 1 along with the generalized frameworks of existing self-adaptive DE algorithms and CoDE. It can be observed from Fig. 1 that the main idea of DESSA is quite different from those of CoDE and existing self-adaptive variants of DE.

3.2. DESSA-CoDE

The newly proposed self-adaptation scheme can be directly incorporated with CoDE, thereby leading to a new self-adaptive DE variant. Algorithm 3 presents the detailed steps of the new algorithm, namely DESSA-CoDE.

DESSA-CoDE uses the same strategies and parameter settings as CoDE. This yields a total of $3 \times 3 = 9$ combinations of strategies and parameter settings. DESSA-CoDE begins with a randomly generated population and evolves it as CoDE does in [14] for $MaxGdb$ generations. At each generation G $(G > MaxGdb)$, each of the 9 combinations is used to create a trial vector for each target vector. As the surrogate model is built to select the most promising trial vector, we consider ranking models in the implementation, which seem more applicable compared to interpolate or regression models when the best individuals need to be selected [34, 35], and use Rank-SVM [36] as the surrogate model, which was also used in [34, 35]. To build a Rank-SVM model S, we select min(k, $|DB|$) nearest evaluated points for each generated trial vector based on the Euclidean distance and combine them together, 80% of which are chosen uniformly as the training set and the remaining 20% form the set for validating the prediction quality. If the prediction accuracy of S is larger than 0.5 (the accuracy of a random approach), the trial vector appears the best according to S is further evaluated with the exact objective function, otherwise a trial vector is randomly selected for evaluation, which will enter the next generation if it is better than the target vector. This process iterates until all FEs are used up.

According to Algorithm 3, DESSA-CoDE can automatically select a promising combination of strategy and parameter setting for each target vector without extra FEs, and thus can be more cost-effective than CoDE.

Algorithm 3 DESSA-CoDE

Input:

the objective function, f;
maximal number of FEs, $MaxEval$;
candidate strategies: "DE/rand/1/bin", "DE/rand/2/bin", and "DE/current-to-rand/1";
candidate parameter settings: [$F = 1.0$, $CR = 0.1$], [$F = 1.0$, $CR = 0.9$], and [$F = 0.8$, $CR = 0.2$];

1: Set $popsize$, $MaxGdb$, $G = 0$, $eval = 0$, $DB = \Phi$
2: Initialize a population $\boldsymbol{P}_G = \{\boldsymbol{x}_{i,G} | i = 1, 2, ..., popsize\}$
3: Archive all $(\boldsymbol{x}_{i,G}, f(\boldsymbol{x}_{i,G}))$ into DB
4: $eval = eval + popsize$
5: **while** $eval < MaxEval$ **do**
6:   **if** $G < MaxGdb$ **then**
7:     Generate $\boldsymbol{P}_{G+1}$ based on $\boldsymbol{P}_G$ as CoDE does
8:   **else**
9:     **for** each $\boldsymbol{x}_{i,G}$ in $\boldsymbol{P}_G$ **do**
10:       Generate nine trial vectors using the nine combinations of strategies and parameter settings
11:       Build a ranking model S based on DB
12:       **if** The prediction accuracy of S is larger than 0.5 **then**
13:         Select the best trial vector according to S (denoted as $\boldsymbol{u}_{i,G}$)
14:       **else**
15:         Randomly select one trial vector as $\boldsymbol{u}_{i,G}$
16:       **end if**
17:       Evaluate $\boldsymbol{u}_{i,G}$ with f
18:       $\boldsymbol{P}_{G+1} = \boldsymbol{P}_{G+1} \bigcup$ Select($\boldsymbol{x}_{i,G}$, $\boldsymbol{u}_{i,G}$)
19:       $eval = eval + 1$
20:     **end for**
21:   **end if**
22:   Archive all exact evaluations into DB (like step 3)
23:   $G = G + 1$
24: **end while**

Output:

the best individual in the current population, $\boldsymbol{x}_{best,G}$;

3.3. Discussion

It should be noted that DESSA-CoDE in this paper serves as an instantiation of DESSA. In general, DESSA can also be combined with any other DE variant that involve multiple search strategies, and thus generating other instantiations of DESSA, which will be further investigated in our experimental section. Moreover, according to Fig. 1, incorporating more strategies and parameter settings into DESSA will only lead to more trial vectors generated for each target vector, while no additional FEs will be induced as all newly generated trial vectors are to be first filtered by the surrogate model. Therefore, DESSA may easily accommodate more strategies and parameter settings. In fact, other ways to select the training set and other modeling techniques can also be used in the implementation of DESSA-CoDE. However, considering that it is the effect of combining DE and surrogate models that is the key in this paper, no more attention will be paid to this in the following sections.

As mentioned above, there exist a few attempts to incorporate surrogate models with DE in literature [26, 27, 28, 29, 30, 31]. In [28], classification models are built to estimate whether offspring are better than their separate parents and the worse ones are prevented from being evaluated with the real fitness function. Based on this work, CRADE was proposed in [29] by incorporating classification and regression techniques to construct a more effective surrogate model. In [26, 27], DE-AEC was proposed based on jDE by generating multiple offspring for each parent and selecting one to compete with the parent according to a surrogate model. The main difference between our work and DE-AEC is that we considers multiple trial vector strategies and parameter settings while DE-AEC considers only one strategy and parameter setting for each target vector. The authors in [31] enhanced DE by generating multiple trial vectors with several trial vector strategies and building surrogate models to select the most promising one. Unlike their work, our work considers different strategies as well as different control parameter settings. In [30], a surrogate model-assisted EPSDE algorithm, SMA-EPSDE, was proposed by generating a competitive trail vector for each target vector. The difference between SMA-EPSDE and DESSA is that SMA-EPSDE stops generating trial vectors for a target vector once obtaining a competitive trial vector according to the surrogate model while DESSA employs surrogate models to select the best one among multiple trial vectors. Furthermore, none of the aforementioned surrogate-assisted DE studies investigate the role of surrogate models in DE from the perspective of self-adaptation, and most of them take into account only one DE variant.

4. Empirical Study

To assess the performance of the new self-adaptation scheme, we have carried out different experiments using a test suite proposed in the CEC2005 special session on real-parameter optimization. The test suite consists of 25 benchmark functions, including:

- unimodal functions f_1-f_5,
- basic multimodal functions f_6-f_{12},
- expanded multimodal functions f_{13}-f_{14}, and

- hybrid composition functions f_{15}-f_{25}.

A detailed description of them can be found in [37]. The focus of this study was to check whether the new self-adaptation scheme can suits CEPs better, and we studied the performance of DESSA-CoDE along this direction. We also studied the efficiency of this new scheme by comparing it with the self-adaptation scheme of SaDE.

The dimensionality of the test functions was set to 30 throughout the experiments. Considering that only limited computational resources are allowable to solve CEPs, all the algorithms in our experiments were assigned with 3000 FEs. For each algorithm, the average and standard deviation of the minimum function error values it can find on each test function over 25 independent runs using 3000 FEs were recorded for measuring its performance. The function error value of a solution equals to the function value of the solution minus the minimal value of the objective function. To make a comparison between one algorithm and another, we conducted the Wilcoxon rank-sum test at a 0.05 significance level.

4.1. Performance of DESSA-CoDE

To assess the efficacy of DESSA-CoDE, performance comparisons have been made between it and eight state-of-the-art DE variants including CoDE, four self-adaptive DE variants (i.e., jDE, SaDE, JADE, and EPSDE), one DE variant with accelerated convergence rate (i.e., DEahcSPX [38]), and two surrogate model-assisted DE variants (i.e., SMA-EPSDE and CRADE). In DEahcSPX, an adaptive local search operation with a hill-climbing heuristic was employed to improve the performance of DE. Also, DESSA-CoDE was compared with one non-DE approach, the standard covariance matrix adaptation evolution strategy (CMA-ES) [39], which is a very efficient and well-known ES.

It should be noted here that the ideas behind CRADE and the DESSA framework are quite different. CRADE, specifically designed for solving CEPs, attempts using surrogate models to check whether an offspring is worthy of real fitness evaluation, and thus prevent wasting fitness evaluations on unpromising offsprings. On the other hand, DESSA builds surrogate models to select the best one among multiple generated trial vectors along with the best strategy and parameter setting so as to generate better offsprings without extra fitness evaluations. In fact, the idea of CRADE can be directly incorporated into DESSA by introducing surrogate models of CRADE to decide whether the trial vectors selected by DESSA should undergo real fitness evaluations. As this work aims to investigate the performance of DE in solving CEPs from the perspective of self-adaption, we focused the experimental studies on testing the effectiveness of surrogate models in this background. Thus, the comparison between CRADE and DESSA-CoDE serves primarily as a reference.

The parameters of all the compared algorithms except SaDE were directly set the same as in their original papers. In [5], the learning period (LP) was set 50 for SaDE, which means the parameter and strategy are adapted after the initial 50 generations. This seems not a best setting when only 3000 FEs are available. To make a fair comparison, we run SaDE 25 times on each test function with 3000 FEs and six different different LPs of 1, 5, 15, 25, 35, 50, and then SaDE with the best performing LP value was selected for comparison. Note that the other parameters of SaDE were set the same as in [5].

Table 1 summarizes the average and standard deviation of the function error values that SaDE obtained with six different LPs. Through the Wilcoxon rank-sum test, we found that, overall, SaDE with LPs of 1 and 5 performed best. So, SaDE with LP of 5 was used for the comparison between SaDE and DESSA-CoDE.

Table 1: Experimental results of SaDE with six different LPs over 25 runs with 3000 FEs on 25 test functions of 30 variables

Func	LP=1 MeanError±StdDev	LP=5 MeanError±StdDev	LP=15 MeanError±StdDev	LP=25 MeanError±StdDev	LP=35 MeanError±StdDev	LP=50 MeanError±StdDev
f_1	5.27e+003±1.67e+003	5.05e+003±1.31e+003	5.09e+003±1.25e+003	5.05e+003±1.09e+003	5.56e+003±7.99e+002	6.32e+003±9.56e+002
f_2	2.22e+004±3.84e+003	2.34e+004±4.69e+003	2.38e+004±4.90e+003	2.32e+004±4.87e+003	2.31e+004±4.04e+003	2.44e+004±3.76e+003
f_3	7.97e+007±2.72e+007	8.76e+007±3.01e+007	9.54e+007±2.99e+007	1.06e+008±3.66e+007	1.12e+008±3.95e+007	1.48e+008±4.10e+007
f_4	2.87e+004±6.15e+003	2.68e+004±4.98e+003	2.89e+004±3.63e+003	2.78e+004±4.19e+003	3.01e+004±5.53e+003	2.95e+004±5.06e+003
f_5	1.73e+004±1.83e+003	1.70e+004±1.41e+003	1.67e+004±1.40e+003	1.73e+004±1.70e+003	1.76e+004±1.83e+003	1.75e+004±1.52e+003
f_6	7.02e+008±2.86e+008	5.02e+008±2.05e+008	6.25e+008±1.99e+008	5.16e+008±2.34e+008	6.59e+008±1.70e+008	8.19e+008±2.52e+008
f_7	1.16e+003±2.27e+002	1.21e+003±2.02e+002	1.08e+003±2.45e+002	1.15e+003±2.78e+002	1.20e+003±3.16e+002	1.37e+003±2.95e+002
f_8	2.12e+001±4.98e-002	2.12e+001±5.33e-002	2.11e+001±6.99e-002	2.11e+001±5.52e-002	2.12e+001±5.48e-002	2.11e+001±6.39e-002
f_9	2.20e+002±1.73e+001	2.29e+002±1.55e+001	2.42e+002±1.97e+001	2.43e+002±1.28e+001	2.49e+002±1.94e+001	2.45e+002±1.83e+001
f_{10}	3.10e+002±2.35e+001	3.02e+002±1.80e+001	3.03e+002±1.92e+001	3.07e+002±2.18e+001	3.08e+002±1.64e+001	3.12e+002±2.16e+001
f_{11}	4.22e+001±1.25e+000	4.27e+001±1.36e+000	4.34e+001±1.31e+000	4.40e+001±1.51e+000	4.34e+001±1.54e+000	4.34e+001±1.70e+000
f_{12}	5.73e+005±7.02e+004	5.51e+005±9.24e+004	5.50e+005±1.09e+005	5.99e+005±1.08e+005	5.86e+005±1.07e+005	6.32e+005±9.88e+004
f_{13}	1.98e+001±2.09e+000	2.06e+001±1.63e+000	2.10e+001±1.14e+000	2.17e+001±2.04e+000	2.17e+001±1.84e+000	2.30e+001±1.32e+000
f_{14}	1.39e+001±1.45e-001	1.39e+001±1.91e-001	1.39e+001±2.11e-001	1.40e+001±1.15e-001	1.40e+001±1.76e-001	1.40e+001±1.86e-001
f_{15}	6.90e+002±8.79e+001	7.01e+002±7.67e+001	7.08e+002±7.87e+001	7.01e+002±7.82e+001	7.19e+002±7.79e+001	7.06e+002±8.64e+001
f_{16}	4.08e+002±6.74e+001	4.09e+002±6.06e+001	3.85e+002±4.69e+001	4.19e+002±6.67e+001	4.10e+002±6.74e+001	4.18e+002±6.78e+001
f_{17}	4.80e+002±7.18e+001	5.04e+002±8.32e+001	4.78e+002±7.70e+001	4.75e+002±7.54e+001	4.80e+002±5.58e+001	5.34e+002±1.06e+002
f_{18}	1.08e+003±1.79e+001	1.07e+003±1.48e+001	1.07e+003±1.84e+001	1.08e+003±1.92e+001	1.08e+003±1.43e+001	1.08e+003±1.86e+001
f_{19}	1.08e+003±2.20e+001	1.07e+003±2.29e+001	1.07e+003±1.74e+001	1.07e+003±1.52e+001	1.08e+003±2.28e+001	1.07e+003±2.63e+001
f_{20}	1.07e+003±2.13e+001	1.08e+003±2.10e+001	1.07e+003±1.62e+001	1.08e+003±1.85e+001	1.07e+003±2.40e+001	1.08e+003±2.31e+001
f_{21}	1.17e+003±7.04e+001	1.17e+003±4.36e+001	1.17e+003±3.70e+001	1.15e+003±7.16e+001	1.17e+003±4.77e+001	1.19e+003±3.42e+001
f_{22}	1.18e+003±4.03e+001	1.20e+003±2.94e+001	1.18e+003±4.18e+001	1.19e+003±3.27e+001	1.18e+003±3.84e+001	1.22e+003±3.65e+001
f_{23}	1.18e+003±4.34e+001	1.17e+003±5.14e+001	1.17e+003±4.58e+001	1.18e+003±3.85e+001	1.17e+003±4.35e+001	1.19e+003±3.72e+001
f_{24}	1.24e+003±6.55e+001	1.23e+003±4.74e+001	1.20e+003±7.22e+001	1.22e+003±5.47e+001	1.25e+003±4.09e+001	1.24e+003±3.98e+001
f_{25}	1.33e+003±6.55e+001	1.29e+003±1.10e+002	1.27e+003±1.02e+002	1.29e+003±5.62e+001	1.30e+003±6.27e+001	1.29e+003±1.11e+002

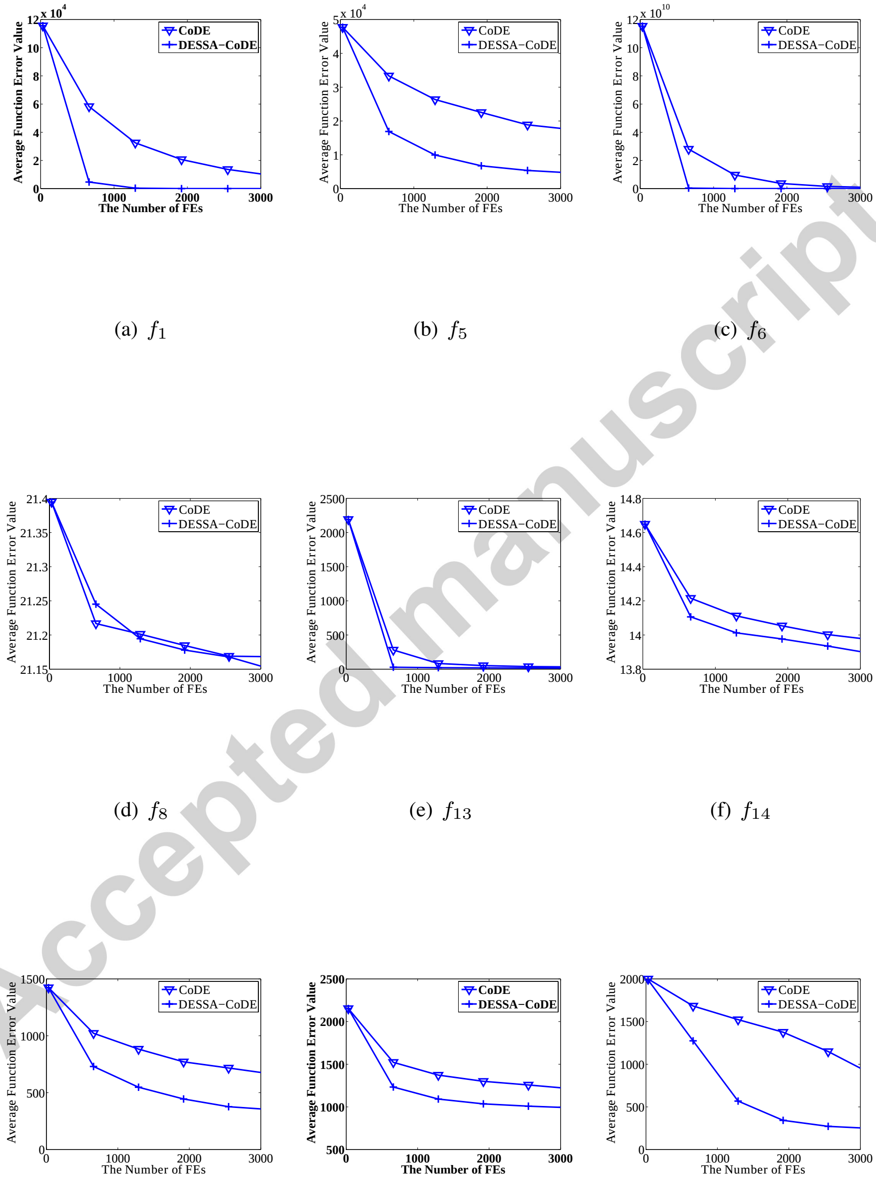
The parameters of DESSA-CoDE were set as: $popsize = 30$, $k = n^2/9$. Note that the population size $popsize$ of DESSA-CoDE was set the same as that of CoDE, and the product of k and the number of the generated trial vectors for each target vector is approximately proportionate to $(n+1)(n+2)/2$, which is the minimum number of data points required to build a quadratic regression model, while k was set to be moderate considering the complexity of training a model. The maximum number of iterations of the SVM learning algorithms was set to $50000\sqrt{n}$. The constraint weights C_i were set to $10^6(\lfloor 0.8k \rfloor - i)^2$, which implies that the cost of constraint violation quadratically increases for the points ranking the top. Here, $\lfloor 0.8k \rfloor$ is the number of training points used to build an Rank-SVM model. Moreover, we employs the RBF kernel function and the kernel width was set to the average distance between training points. It is worth noting that, since we aim at a suitable self-adaptive scheme for solving CEPs, i.e., problems that may cost from minutes to hours of computational time per evaluation, the overhead for identifying the nearest k points and building surrogate models are considered to be insignificant. The setting of $MaxGdb$ indicates how many data points are accumulated before a surrogate model is involved in the search process. In our experimental study, we run DESSA-CoDE 25 times on each test function with 3000 FEs and six different $MaxGdb$ values of 0, 1, 2, 3, 5, 8. Table 2 summarizes the average and standard deviation of the function error values that DESSA-CoDE obtained with six different $MaxGdb$ values. Through the Wilcoxon rank-sum test, it is found that, DESSA-CoDE works best with $MaxGdb$ value of 0.

Table 2: Experimental results of DESSA-CoDE with six different $MaxGdb$ over 25 runs with 3000 FEs on 25 test functions of 30 variables

Func	$MaxGdb$=0 MeanError±StdDev	$MaxGdb$=1 MeanError±StdDev	$MaxGdb$=2 MeanError±StdDev	$MaxGdb$=3 MeanError±StdDev	$MaxGdb$=5 MeanError±StdDev	$MaxGdb$=8 MeanError±StdDev
f_1	3.63e-001±1.56e-001	6.27e-001±4.66e-001	6.87e-001±1.81e-001	1.18e+000±5.97e-001	1.74e+000±3.89e-001	3.73e+000±1.32e+000
f_2	9.39e+003±2.17e+003	1.17e+004±2.84e+003	1.06e+004±2.19e+003	1.04e+004±4.13e+003	1.01e+004±3.32e+003	1.25e+004±4.58e+003
f_3	1.41e+007±5.70e+006	1.60e+007±7.60e+006	1.66e+007±8.11e+006	1.72e+007±5.57e+006	2.35e+007±1.37e+007	2.77e+007±1.18e+007
f_4	2.24e+004±5.68e+003	2.05e+004±6.82e+003	2.36e+004±5.79e+003	2.23e+004±5.10e+003	1.99e+004±5.53e+003	2.59e+004±9.81e+003
f_5	4.81e+003±7.20e+002	4.47e+003±1.00e+003	5.11e+003±7.35e+002	5.16e+003±8.78e+002	5.42e+003±8.32e+002	5.60e+003±9.65e+002
f_6	4.73e+003±6.82e+003	5.12e+003±3.84e+003	6.60e+003±4.79e+003	3.91e+003±2.29e+003	3.61e+003±2.05e+003	1.12e+004±5.88e+003
f_7	1.42e+001±8.61e+000	1.60e+001±9.19e+000	1.54e+001±7.48e+000	1.93e+001±8.45e+000	2.30e+001±5.78e+000	3.29e+001±1.59e+001
f_8	2.12e+001±6.53e-002	2.12e+001±5.17e-002	2.12e+001±5.39e-002	2.11e+001±4.00e-002	2.11e+001±6.53e-002	2.12e+001±6.62e-002
f_9	1.66e+002±9.71e+000	1.79e+002±5.11e+000	1.66e+002±1.35e+001	1.80e+002±1.34e+001	1.69e+002±1.75e+001	1.83e+002±1.38e+001
f_{10}	2.43e+002±2.15e+001	2.45e+002±1.49e+001	2.48e+002±1.69e+001	2.46e+002±2.12e+001	2.44e+002±2.52e+001	2.42e+002±1.27e+001
f_{11}	4.18e+001±1.57e+000	4.16e+001±1.35e+000	4.12e+001±1.53e+000	4.25e+001±9.44e-001	4.12e+001±1.79e+000	4.20e+001±1.85e+000
f_{12}	7.39e+004±5.68e+004	5.86e+004±2.81e+004	1.09e+005±7.04e+004	1.20e+005±1.06e+005	8.38e+004±4.55e+004	1.48e+005±1.30e+005
f_{13}	1.62e+001±1.19e+000	1.61e+001±1.72e+000	1.78e+001±1.76e+000	1.74e+001±1.50e+000	1.67e+001±1.66e+000	1.75e+001±1.76e+000
f_{14}	1.39e+001±1.48e-001	1.39e+001±1.69e-001	1.40e+001±1.26e-001	1.40e+001±2.06e-001	1.39e+001±1.40e-001	1.40e+001±1.08e-001
f_{15}	3.55e+002±5.98e+001	3.59e+002±4.25e+001	3.73e+002±5.14e+001	3.37e+002±9.04e+001	4.24e+002±8.33e+001	4.23e+002±6.44e+001
f_{16}	3.19e+002±8.15e+001	3.10e+002±6.45e+001	3.01e+002±5.38e+001	3.24e+002±8.52e+001	2.82e+002±5.47e+001	2.81e+002±4.64e+001
f_{17}	3.48e+002±6.28e+001	3.96e+002±8.83e+001	3.63e+002±5.95e+001	3.91e+002±8.58e+001	3.73e+002±7.05e+001	3.58e+002±5.62e+001
f_{18}	9.15e+002±3.27e+000	9.16e+002±2.57e+000	9.15e+002±3.31e+000	9.06e+002±3.17e+001	9.17e+002±3.74e+000	9.21e+002±3.15e+000
f_{19}	9.15e+002±2.34e+000	9.16e+002±2.44e+000	9.15e+002±3.08e+000	9.16e+002±2.96e+000	9.17e+002±2.26e+000	9.20e+002±3.90e+000
f_{20}	9.16e+002±3.44e+000	9.15e+002±2.74e+000	9.14e+002±2.16e+000	9.15e+002±2.92e+000	9.18e+002±4.33e+000	9.18e+002±3.86e+000
f_{21}	5.00e+002±1.89e-001	5.01e+002±8.80e-001	5.01e+002±5.58e-001	5.01e+002±3.52e-001	5.01e+002±5.21e-001	5.02e+002±1.01e+000
f_{22}	9.97e+002±2.69e+001	9.93e+002±3.44e+001	9.80e+002±2.83e+001	9.84e+002±2.97e+001	9.85e+002±2.62e+001	1.02e+003±3.66e+001
f_{23}	5.35e+002±1.33e+000	5.34e+002±3.42e-001	5.35e+002±2.08e+000	5.85e+002±1.48e+002	5.36e+002±1.53e+000	5.41e+002±8.19e+000
f_{24}	2.03e+002±3.76e+000	2.02e+002±1.01e+000	2.03e+002±2.75e+000	2.03e+002±1.93e+000	2.13e+002±1.07e+001	2.15e+002±2.17e+001
f_{25}	2.52e+002±1.54e+001	2.65e+002±5.02e+001	2.55e+002±1.11e+001	2.56e+002±2.69e+001	2.59e+002±1.17e+001	2.70e+002±3.11e+001

Table 3: Experimental results of CoDE and DESSA-CoDE over 25 runs with 3000 FEs on 25 test functions of 30 variables, $+$, $-$, and $\approx$ denote that the result of CoDE is better than, worse than, and comparable to that of DESSA-CoDE, respectively

Func	CoDE MeanError±StdDev	DESSA-CoDE MeanError±StdDev
f_1	1.02e+004±2.92e+003 −	3.63e-001±1.56e-001
f_2	3.84e+004±6.36e+003 −	9.39e+003±2.17e+003
f_3	2.07e+008±6.65e+007 −	1.41e+007±5.70e+006
f_4	4.79e+004±8.80e+003 −	2.24e+004±5.68e+003
f_5	1.81e+004±1.74e+003 −	4.81e+003±7.20e+002
f_6	9.03e+008±4.77e+008 −	4.73e+003±6.82e+003
f_7	2.02e+003±4.86e+002 −	1.42e+001±8.61e+000
f_8	2.12e+001±4.35e-002 ≈	2.12e+001±6.53e-002
f_9	2.43e+002±1.76e+001 −	1.66e+002±9.71e+000
f_{10}	3.43e+002±2.56e+001 −	2.43e+002±2.15e+001
f_{11}	4.33e+001±1.45e+000 −	4.18e+001±1.57e+000
f_{12}	7.48e+005±1.14e+005 −	7.39e+004±5.68e+004
f_{13}	3.25e+001±4.88e+000 −	1.62e+001±1.19e+000
f_{14}	1.40e+001±1.97e-001 −	1.39e+001±1.48e-001
f_{15}	6.79e+002±7.37e+001 −	3.55e+002±5.98e+001
f_{16}	4.12e+002±5.30e+001 −	3.19e+002±8.15e+001
f_{17}	4.56e+002±4.83e+001 −	3.48e+002±6.28e+001
f_{18}	1.05e+003±2.03e+001 −	9.15e+002±3.27e+000
f_{19}	1.06e+003±2.03e+001 −	9.15e+002±2.34e+000
f_{20}	1.04e+003±1.80e+001 −	9.16e+002±3.44e+000
f_{21}	1.18e+003±4.34e+001 −	5.00e+002±1.89e-001
f_{22}	1.22e+003±3.84e+001 −	9.97e+002±2.69e+001
f_{23}	1.20e+003±3.83e+001 −	5.35e+002±1.33e+000
f_{24}	1.19e+003±5.63e+001 −	2.03e+002±3.76e+000
f_{25}	9.30e+002±2.78e+002 −	2.52e+002±1.54e+001
−	24	
+	0	
≈	1	



(a) f_1 (b) f_5 (c) f_6

(d) f_8 (e) f_{13} (f) f_{14}

(g) f_{15} (h) f_{22} (i) f_{25}

Figure 2: Evolutionary curves of CoDE and DESSA-CoDE

Table 4: Experimental results of jDE, SaDE, JADE, EPSDE, and DESSA-CoDE over 25 runs with 3000 FEs on 25 test functions of 30 variables, $+$, $-$, and $\approx$ denote that the result of the corresponding algorithm is better than, worse than, and comparable to that of DESSA-CoDE, respectively

Func	jDE MeanError±StdDev	SaDE MeanError±StdDev	JADE MeanError±StdDev	EPSDE MeanError±StdDev	DESSA-CoDE MeanError±StdDev
f_1	1.71e+004±3.25e+003 −	5.05e+003±1.31e+003 −	4.01e+003±7.13e+002 −	3.09e+003±1.42e+003 −	3.63e-001±1.56e-001
f_2	6.15e+004±9.33e+003 −	2.34e+004±4.69e+003 −	3.74e+004±5.47e+003 −	6.21e+004±1.60e+004 −	9.39e+003±2.17e+003
f_3	4.41e+008±9.80e+007 −	8.76e+007±3.01e+007 −	1.91e+008±3.96e+007 −	4.03e+008±1.94e+008 −	1.41e+007±5.70e+006
f_4	7.15e+004±1.18e+004 −	2.68e+004±4.98e+003 −	5.04e+004±9.77e+003 −	8.27e+004±2.52e+004 −	2.24e+004±5.68e+003
f_5	2.00e+004±1.90e+003 −	1.70e+004±1.41e+003 −	1.24e+004±1.13e+003 −	1.76e+004±4.05e+003 −	4.81e+003±7.20e+002
f_6	3.52e+009±9.66e+008 −	5.02e+008±2.05e+008 −	1.86e+008±7.10e+007 −	3.02e+008±3.04e+008 −	4.73e+003±6.82e+003
f_7	2.89e+003±3.32e+002 −	1.21e+003±2.02e+002 −	7.12e+002±1.16e+002 −	6.78e+002±2.03e+002 −	1.42e+001±8.61e+000
f_8	2.11e+001±5.11e-002 ≈	2.12e+001±5.33e-002 ≈	2.12e+001±4.99e-002 ≈	2.11e+001±4.95e-002 ≈	2.12e+001±6.53e-002
f_9	2.76e+002±1.81e+001 −	2.29e+002±1.55e+001 −	2.28e+002±1.53e+001 −	2.45e+002±2.12e+001 −	1.66e+002±9.71e+000
f_{10}	3.82e+002±2.59e+001 −	3.02e+002±1.80e+001 −	2.89e+002±1.68e+001 −	3.03e+002±1.93e+001 −	2.43e+002±2.15e+001
f_{11}	4.30e+001±1.72e+000 −	4.27e+001±1.36e+000 ≈	4.39e+001±1.28e+000 −	4.43e+001±1.14e+000 −	4.18e+001±1.57e+000
f_{12}	8.58e+005±1.42e+005 −	5.51e+005±9.24e+004 −	6.79e+005±8.53e+004 −	9.02e+005±1.22e+005 −	7.39e+004±5.68e+004
f_{13}	7.31e+001±1.81e+001 −	2.06e+001±1.63e+000 −	2.64e+001±1.94e+000 −	3.20e+001±1.19e+001 −	1.62e+001±1.19e+000
f_{14}	1.40e+001±1.08e-001 −	1.39e+001±1.91e-001 ≈	1.39e+001±1.41e-001 ≈	1.42e+001±1.74e-001 −	1.39e+001±1.48e-001
f_{15}	7.84e+002±7.97e+001 −	7.01e+002±7.67e+001 −	6.52e+002±9.54e+001 −	8.63e+002±3.90e+001 −	3.55e+002±5.98e+001
f_{16}	4.87e+002±5.20e+001 −	4.09e+002±6.06e+001 −	3.54e+002±5.08e+001 −	4.17e+002±1.00e+002 −	3.19e+002±8.15e+001
f_{17}	5.61e+002±6.48e+001 −	5.04e+002±8.32e+001 −	3.96e+002±8.06e+001 −	4.96e+002±8.11e+001 −	3.48e+002±6.28e+001
f_{18}	1.10e+003±3.19e+001 −	1.07e+003±1.48e+001 −	9.69e+002±1.17e+001 −	9.49e+002±3.90e+001 −	9.15e+002±3.27e+000
f_{19}	1.09e+003±3.21e+001 −	1.07e+003±2.29e+001 −	9.72e+002±9.87e+000 −	9.46e+002±3.56e+001 −	9.15e+002±2.34e+000
f_{20}	1.10e+003±2.06e+001 −	1.08e+003±2.10e+001 −	9.69e+002±1.01e+001 −	9.40e+002±2.31e+001 −	9.16e+002±3.44e+000
f_{21}	1.23e+003±1.91e+001 −	1.17e+003±4.36e+001 −	1.05e+003±8.38e+001 −	9.70e+002±3.21e+001 −	5.00e+002±1.89e-001
f_{22}	1.28e+003±5.38e+001 −	1.20e+003±2.94e+001 −	1.11e+003±3.21e+001 −	8.22e+002±1.52e+002 +	9.97e+002±2.69e+001
f_{23}	1.24e+003±1.49e+001 −	1.17e+003±5.14e+001 −	1.03e+003±7.64e+001 −	9.74e+002±3.47e+001 −	5.35e+002±1.33e+000
f_{24}	1.27e+003±2.87e+001 −	1.23e+003±4.74e+001 −	1.01e+003±6.09e+001 −	4.05e+002±1.07e+002 −	2.03e+002±3.76e+000
f_{25}	1.45e+003±3.58e+001 −	1.29e+003±1.10e+002 −	1.02e+003±2.13e+002 −	5.39e+002±2.37e+002 −	2.52e+002±1.54e+001
−	24	22	23	23	
+	0	0	0	1	
≈	1	3	2	1	

Table 5: Experimental results of DEahcSPX, SMA-EPSDE, CRADE, CMA-ES, and DESSA-CoDE over 25 runs with 3000 FEs on 25 test functions of 30 variables, $+$, $-$, and $\approx$ denote that the result of the corresponding algorithm is better than, worse than, and comparable to that of DESSA-CoDE, respectively

Func	DEahcSPX MeanError±StdDev	SMA-EPSDE MeanError±StdDev	CRADE MeanError±StdDev	CMA-ES MeanError±StdDev	DESSA-CoDE MeanError±StdDev
f_1	8.63e+003±2.58e+003 −	1.44e+004±2.44e+003 −	2.34e+000±5.56e-001 −	2.36e+004±5.49e+003 −	3.63e-001±1.56e-001
f_2	2.71e+004±3.68e+003 −	3.47e+004±5.43e+003 −	1.89e+004±5.75e+003 −	2.71e+004±3.13e+004 −	9.39e+003±2.17e+003
f_3	1.73e+008±7.25e+007 −	2.40e+008±7.36e+007 −	7.38e+007±1.97e+007 −	4.76e+008±3.89e+008 −	1.41e+007±5.70e+006
f_4	3.74e+004±6.10e+003 −	5.23e+004±8.27e+003 −	2.52e+004±6.76e+003 ≈	2.52e+006±2.91e+006 −	2.24e+004±5.68e+003
f_5	1.81e+004±2.36e+003 −	2.12e+004±2.37e+003 −	4.03e+003±7.02e+002 +	2.27e+004±3.42e+003 −	4.81e+003±7.20e+002
f_6	8.35e+008±4.43e+008 −	2.17e+009±7.21e+008 −	1.15e+007±7.14e+006 −	3.86e+009±2.74e+009 −	4.73e+003±6.82e+003
f_7	2.28e+003±5.21e+002 −	2.46e+003±3.87e+002 −	1.19e+001±6.71e+000 ≈	2.13e+000±7.21e-001 +	1.42e+001±8.61e+000
f_8	2.12e+001±4.94e-002 ≈	2.11e+001±8.20e-002 ≈	2.11e+001±7.38e-002 ≈	2.15e+001±9.80e-002 −	2.12e+001±6.53e-002
f_9	2.84e+002±3.14e+001 −	3.02e+002±1.78e+001 −	1.87e+002±2.95e+001 −	4.60e+002±8.90e+001 −	1.66e+002±9.71e+000
f_{10}	3.19e+002±3.64e+001 −	3.34e+002±2.76e+001 −	2.18e+002±1.91e+001 +	3.47e+002±6.80e+001 −	2.43e+002±2.15e+001
f_{11}	4.36e+001±1.40e+000 −	4.06e+001±1.61e+000 +	3.83e+001±5.14e+000 +	4.05e+001±1.37e+001 ≈	4.18e+001±1.57e+000
f_{12}	8.17e+005±1.70e+005 −	6.18e+005±2.65e+005 −	1.47e+006±2.22e+005 −	2.31e+005±4.13e+005 ≈	7.39e+004±5.68e+004
f_{13}	2.70e+001±2.51e+000 −	1.74e+001±1.78e+000 −	1.93e+001±1.61e+000 −	1.88e+001±4.84e+000 −	1.62e+001±1.19e+000
f_{14}	1.39e+001±2.02e-001 ≈	1.40e+001±1.44e-001 −	1.40e+001±1.07e-001 −	1.47e+001±2.29e-001 −	1.39e+001±1.48e-001
f_{15}	6.43e+002±8.80e+001 −	6.37e+002±9.18e+001 −	3.59e+002±6.25e+001 ≈	8.42e+002±1.60e+002 −	3.55e+002±5.98e+001
f_{16}	3.94e+002±6.29e+001 −	3.60e+002±5.75e+001 −	2.88e+002±9.25e+001 ≈	7.07e+002±2.17e+002 −	3.19e+002±8.15e+001
f_{17}	4.67e+002±7.19e+001 −	4.67e+002±7.66e+001 −	3.51e+002±1.20e+002 ≈	8.26e+002±4.89e+002 −	3.48e+002±6.28e+001
f_{18}	1.06e+003±2.80e+001 −	1.09e+003±2.72e+001 −	9.12e+002±2.76e+000 +	1.12e+003±3.96e+001 −	9.15e+002±3.27e+000
f_{19}	1.07e+003±3.17e+001 −	1.08e+003±2.43e+001 −	9.14e+002±3.29e+000 +	1.10e+003±4.44e+001 −	9.15e+002±2.34e+000
f_{20}	1.07e+003±2.34e+001 −	1.09e+003±2.28e+001 −	9.13e+002±4.33e+000 ≈	1.13e+003±4.48e+001 −	9.16e+002±3.44e+000
f_{21}	1.16e+003±4.58e+001 −	1.16e+003±7.03e+001 −	5.45e+002±1.69e+002 −	1.26e+003±2.98e+001 −	5.00e+002±1.89e-001
f_{22}	1.21e+003±5.22e+001 −	1.21e+003±4.89e+001 −	9.54e+002±2.22e+001 +	1.34e+003±9.58e+001 −	9.97e+002±2.69e+001
f_{23}	1.19e+003±4.37e+001 −	1.19e+003±4.12e+001 −	5.62e+002±1.06e+002 ≈	1.26e+003±2.67e+001 −	5.35e+002±1.33e+000
f_{24}	1.14e+003±6.02e+001 −	1.20e+003±9.07e+001 −	2.01e+002±5.23e-001 ≈	1.28e+003±5.42e+001 −	2.03e+002±3.76e+000
f_{25}	8.24e+002±3.14e+002 −	1.37e+003±4.24e+001 −	2.45e+002±9.34e+000 ≈	2.20e+002±2.04e+001 +	2.52e+002±1.54e+001
−	23	23	9	21	
+	0	1	6	2	
≈	2	1	10	2	

Tables 3, 4 and 5 summarize the average and standard deviation of the function error values obtained by the 10 algorithms on all the test functions. The results of the corresponding Wilcoxon rank-sum tests are presented in the last three rows of the tables.

As can be seen from the last three rows of Table 3, DESSA-CoDE outperformed CoDE on 24 test functions while CoDE failed to surpass DESSA-CoDE on any test function. Furthermore, when looking at the evolution curves of CoDE and DESSA-CoDE, it can be observed that the evolution curves of DESSA-CoDE always lie beneath CoDE on almost all the test functions. Fig. 2 shows the evolution curves of CoDE and DESSA-CoDE on some representative test functions. This substantiates our claim that DESSA-CoDE is more cost-effective than CoDE.

By comparing DESSA-CoDE to the four self-adaptive DE variants (i.e., CoDE, jDE, SaDE, JADE, and EPSDE), we have found that DESSA-CoDE overall performed better than them. Specifically, it can be seen from Table 4 that DESSA-CoDE obtained better solutions than each of jDE, SaDE, JADE and EPSDE on more than 20 test functions, while only EPSDE outperformed DESSA-CoDE on only 1 test function. When compared to DEahcSPX, DESSA-CoDE achieved better results than it on 23 test functions, while DEahcSPX did not outperform DESSA-CoDE on any test function. Moreover, looking at the the last three rows of Table 5, DESSA-CoDE even showed competitive performance in comparison with SMA-EPSDE and CMA-ES, and comparable results in contrast to CRADE.

4.2. Comparison with the self-adaptation scheme of SaDE

According to the above comparisons, DESSA-CoDE exhibited overall better performance than the compared four self-adaptive DE variants. However, as DESSA-CoDE applies a different set of search strategies compared to the four self-adaptive DE variants, the observation can not clearly establish that the newly proposed self-adaptation scheme is better than the existing self-adaptation schemes. In order to clearly evaluate the potential of the new scheme, we further compared it with the self-adaptation scheme applied in SaDE. In this comparison, we embedded the new scheme into SaDE to adapt the trial vector generation strategy instead of the original self-adaptation scheme and compared the resulted algorithm, DESSA-SaDE, with SaDE. Specifically, in DESSA-SaDE, four trial vectors are generated with four trial vector generation strategies that are used in SaDE and a Rank-SVM model is built to select the most promising one for each target vector, while the rest of the algorithm is exactly the same as SaDE. The parameter setting of SaDE is the same as in Section 4.1. The parameters of DESSA-SaDE were set as: $popsize = 50$, $MaxGdb = 0$. As the number of generated trial vectors for each target vector in DESSA-SaDE is 4, k is set to $n^2/4$ instead of $n^2/9$. Note that the population size of DESSA-SaDE was set the same as that of SaDE.

Table 6 demonstrates the average and standard deviation of the best function error values achieved by SaDE and DESSA-SaDE on each test function over 25 independent runs using 3000 FEs and the results of the Wilcoxon rank-sum tests conducted to compare them. From the comparison results in the last three rows of Table 6, it can be seen that DESSA-SaDE overall gives better results than SaDE.

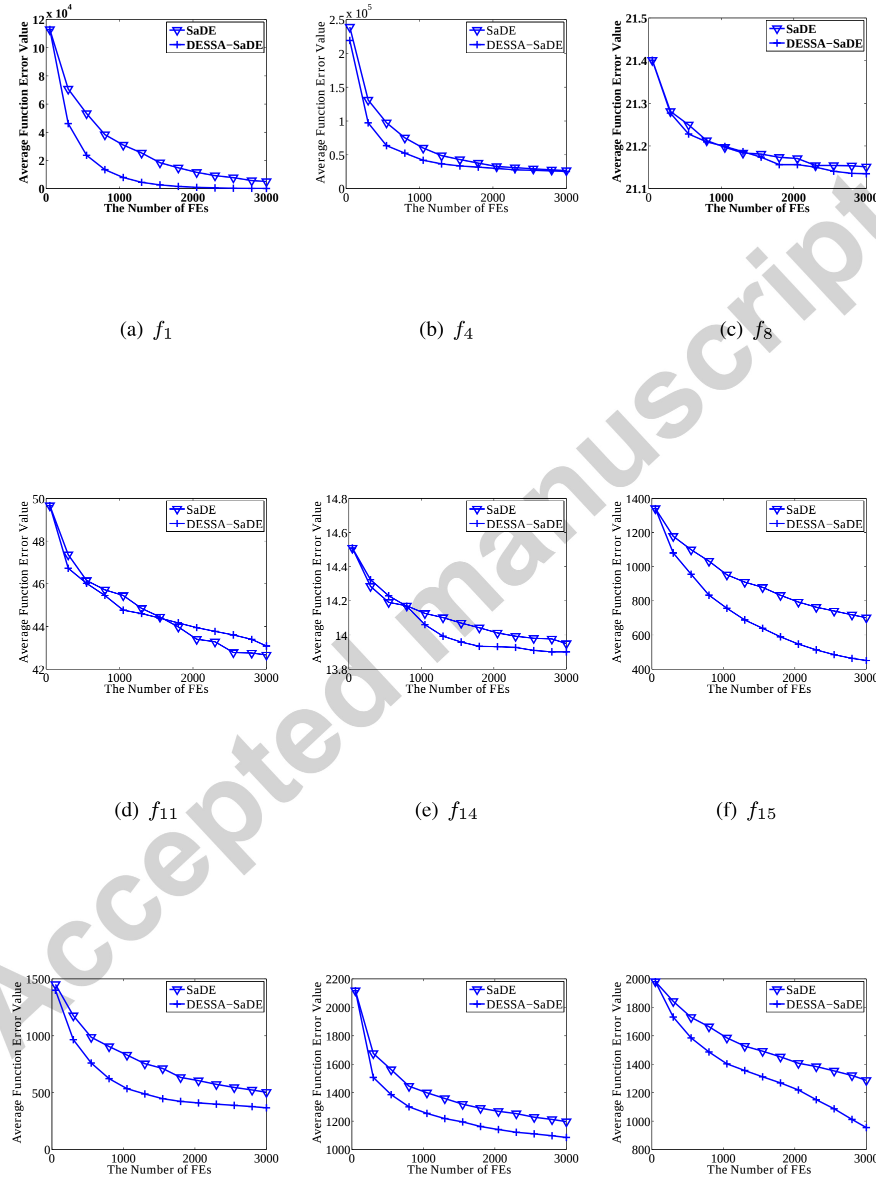
Table 6: Experimental results of SaDE and DESSA-SaDE over 25 runs with 3000 FEs on 25 test functions of 30 variables, $+$, $-$, and $\approx$ denote that the result of SaDE is better than, worse than, and comparable to that of DESSA-SaDE, respectively

Func	SaDE MeanError±StdDev	DESSA-SaDE MeanError±StdDev
f_1	5.05e+003±1.31e+003 $-$	9.55e+001±3.97e+001
f_2	2.34e+004±4.69e+003 $-$	1.47e+004±2.77e+003
f_3	8.76e+007±3.01e+007 $-$	3.69e+007±1.20e+007
f_4	2.68e+004±4.98e+003 $\approx$	2.53e+004±5.42e+003
f_5	1.70e+004±1.41e+003 $-$	7.95e+003±7.91e+002
f_6	5.02e+008±2.05e+008 $-$	4.07e+006±4.16e+006
f_7	1.21e+003±2.02e+002 $-$	1.88e+002±5.39e+001
f_8	2.12e+001±5.33e-002 $\approx$	2.11e+001±6.91e-002
f_9	2.29e+002±1.55e+001 $-$	2.16e+002±1.16e+001
f_{10}	3.02e+002±1.80e+001 $-$	2.56e+002±2.06e+001
f_{11}	4.27e+001±1.36e+000 $\approx$	4.31e+001±1.36e+000
f_{12}	5.51e+005±9.24e+004 $-$	1.54e+005±9.21e+004
f_{13}	2.06e+001±1.63e+000 $-$	1.92e+001±1.57e+000
f_{14}	1.39e+001±1.91e-001 $\approx$	1.39e+001±1.76e-001
f_{15}	7.01e+002±7.67e+001 $-$	4.50e+002±5.69e+001
f_{16}	4.09e+002±6.06e+001 $-$	2.85e+002±2.75e+001
f_{17}	5.04e+002±8.32e+001 $-$	3.67e+002±9.11e+001
f_{18}	1.07e+003±1.48e+001 $-$	9.37e+002±4.11e+001
f_{19}	1.07e+003±2.29e+001 $-$	9.55e+002±1.22e+001
f_{20}	1.08e+003±2.10e+001 $-$	9.45e+002±3.67e+001
$-$	21	
$+$	0	
$\approx$	4	

In addition to the quality of the final solution, the evolution curves of SaDE and DESSA-SaDE on some representative test functions are presented in Fig. 3. The Fig. 3 shows that the evolution curves on almost all the test functions obtained by DESSA-SaDE lie beneath their respective ones obtained by SaDE in the whole search process.

Furthermore, the dynamics of the true rank of the trial vector that was selected by the adaptation scheme in the search process of both SaDE and DESSA-SaDE are plotted in Figs. 4, 5 and 6, where the X-axis represents the number of generations and the Y-axis represents the average rank of the selected trial vectors. Note that, if the selected trial vector is actually the best one among all the generated trail vectors, its true rank is 1. For each generation, the average rank of the selected trial vectors is calculated by averaging over all selected trial vectors first and then averaging over 25 runs. From Figs. 4, 5 and 6, it can be seen that both the self-adaptation scheme of SaDE and that of DESSA-CoDE can not play positive roles in the optimization of f_8 and f_{14}. Except f_8 and f_{14}, the curve of SaDE on each test function shows a general descending trend. Nevertheless, SaDE failed to select good trial vectors in almost the first half of the whole search process as the average rank of the selected trial vectors in such a period showed a slight fluctuation around 2.5, which is the expected rank that can be obtained by the random selection method, thereby substantiating the claim that existing self-adaptation schemes may not function effectively within a small generations. In contrast, the self-adaptation scheme of DESSA-SaDE can select significantly better trial vectors in the early stage even the whole search process one the other 23 test functions. Overall, the newly proposed self-adaptation scheme performed better than that of SaDE, and thus is more appropriate for CEPs.

All the above observations from Table 6 and Figs. 3- 6 constituently establish the newly proposed scheme as a competitive self-adaptation scheme for DE to solve CEPs.

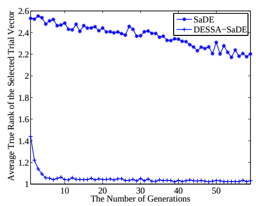


(a) f_1 (b) f_4 (c) f_8

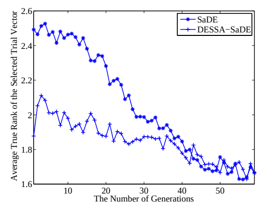
(d) f_{11} (e) f_{14} (f) f_{15}

(g) f_{17} (h) f_{22} (i) f_{25}

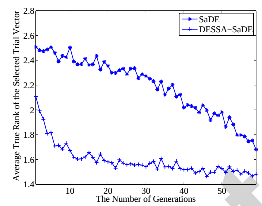
Figure 3: Evolutionary curves of SaDE and DESSA-SaDE



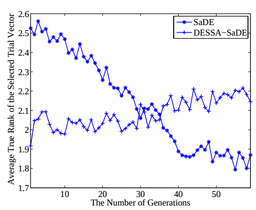
(a) f_1



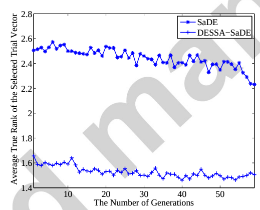
(b) f_2



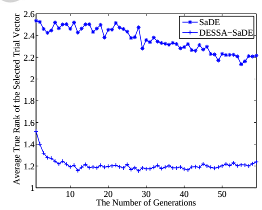
(c) f_3



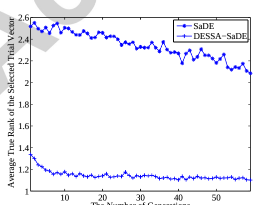
(d) f_4



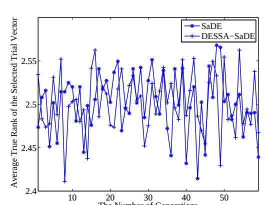
(e) f_5



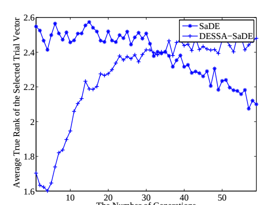
(f) f_6



(g) f_7

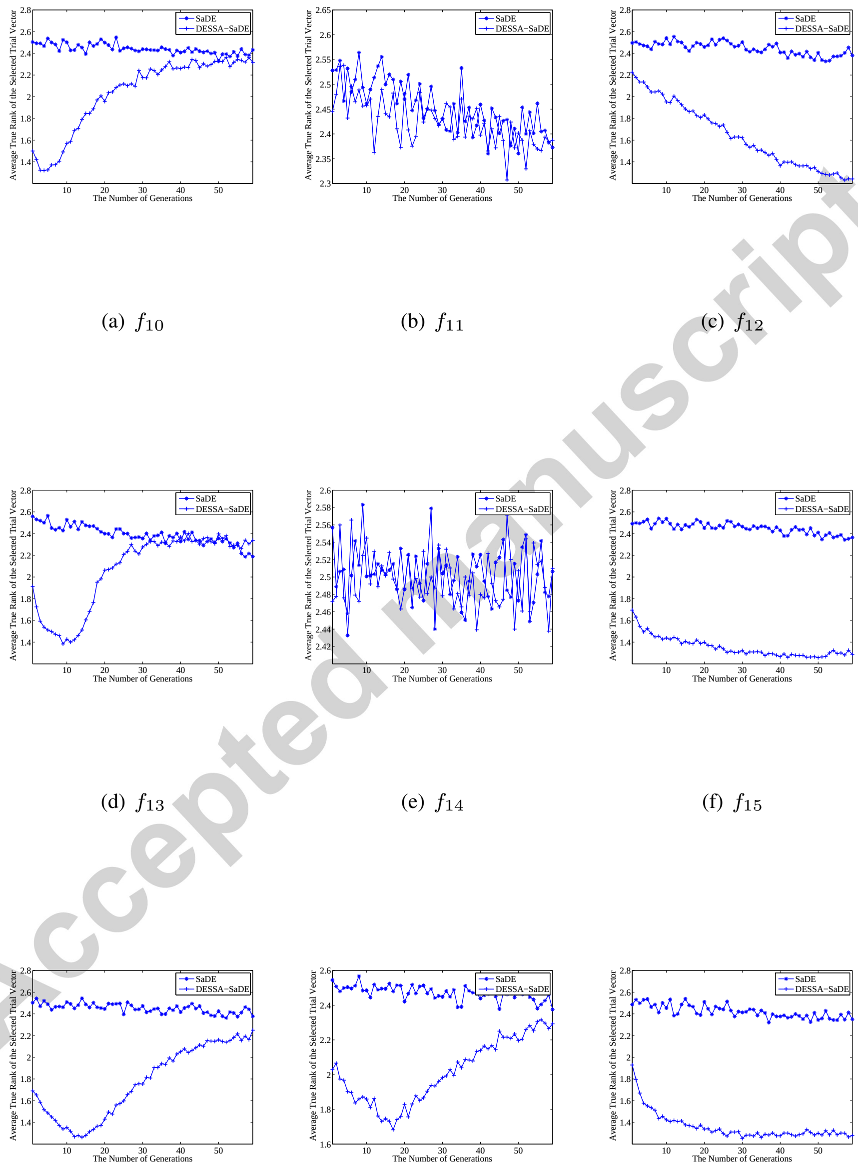


(h) f_8



(i) f_9

Figure 4: Change curves of the true rank of the selected trial vectors in the search process of SaDE and DESSA-SaDE on $f_1 - f_9$

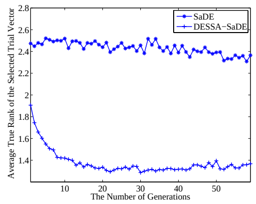


(a) f_{10} (b) f_{11} (c) f_{12}

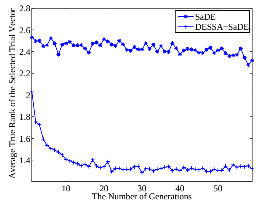
(d) f_{13} (e) f_{14} (f) f_{15}

(g) f_{16} (h) f_{17} (i) f_{18}

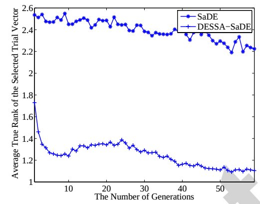
Figure 5: Change curves of the true rank of the selected trial vectors in the search process of SaDE and DESSA-SaDE on $f_{10} - f_{18}$



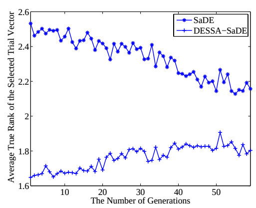
(a) f_{19}



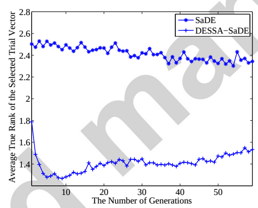
(b) f_{20}



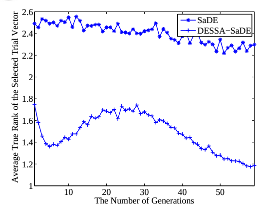
(c) f_{21}



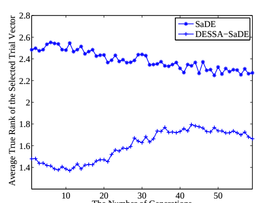
(d) f_{22}



(e) f_{23}



(f) f_{24}



(g) f_{25}

Figure 6: Change curves of the true rank of the selected trial vectors in the search process of SaDE and DESSA-SaDE on $f_{19} - f_{25}$

4.3. More strategies and parameter settings for DESSA

In Section 3, it has been stated that DESSA may easily accommodate more strategies and parameter settings. Therefore, we also conducted some experiments to check whether the performance of DESSA can be further boosted by incorporating more strategies and parameter settings.

First, we formed a new instantiation of DESSA by introducing another strategy and two other parameter settings into DESSA-CoDE, which is referred to as DESSA-CoDE*. As none of the strategies used in DESSA-CoDE rely on the best solution found so far, we selected DE/target-to-best/2 as the added strategy. The two added parameter settings are: $[F = 0.5, CR = 0.9]$ and $[F = 0.5, CR = 0.3]$, which are commonly suggested settings of F and CR [8, 11, 13], while suitably considering the parameter settings of DESSA-CoDE. Then, performance comparisons were made between DESSA-CoDE and DESSA-CoDE* on the 25 test functions.

Table 7: Experimental results of DESSA-CoDE and DESSA-CoDE* over 25 runs with 3000 FEs on 25 test functions of 30 variables, $+$, $-$, and $\approx$ denote that the result of DESSA-CoDE is better than, worse than, and comparable to that of DESSA-CoDE*, respectively

Func	DESSA-CoDE MeanError±StdDev	DESSA-CoDE* MeanError±StdDev
f_1	3.63e-001±1.56e-001 $-$	7.26e-005±6.15e-005
f_2	9.39e+003±2.17e+003 $-$	5.31e+003±8.32e+002
f_3	1.41e+007±5.70e+006 $-$	1.09e+007±5.98e+006
f_4	2.24e+004±5.68e+003 $\approx$	2.57e+004±6.90e+003
f_5	4.81e+003±7.20e+002 $\approx$	4.97e+003±1.05e+003
f_6	4.73e+003±6.82e+003 $\approx$	3.89e+003±4.84e+003
f_7	1.42e+001±8.61e+000 $\approx$	1.04e+001±6.69e+000
f_8	2.12e+001±6.53e-002 $\approx$	2.12e+001±4.48e-002
f_9	1.66e+002±9.71e+000 $\approx$	1.70e+002±1.80e+001
f_{10}	2.43e+002±2.15e+001 $\approx$	2.37e+002±1.63e+001
f_{11}	4.18e+001±1.57e+000 $\approx$	4.15e+001±2.50e+000
f_{12}	7.39e+004±5.68e+004 $-$	3.66e+004±2.51e+004
f_{13}	1.62e+001±1.19e+000 $\approx$	1.65e+001±2.22e+000
f_{14}	1.39e+001±1.48e-001 $\approx$	1.39e+001±1.46e-001
f_{15}	3.55e+002±5.98e+001 $\approx$	3.48e+002±6.14e+001
f_{16}	3.19e+002±8.15e+001 $\approx$	3.25e+002±9.03e+001
f_{17}	3.48e+002±6.28e+001 $\approx$	3.89e+002±1.13e+002
f_{18}	9.15e+002±3.27e+000 $\approx$	9.15e+002±5.34e+000
f_{19}	9.15e+002±2.34e+000 $\approx$	9.16e+002±5.94e+000
f_{20}	9.16e+002±3.44e+000 $\approx$	9.17e+002±6.02e+000
f_{21}	5.00e+002±1.89e-001 $+$	5.25e+002±8.63e+001
f_{22}	9.97e+002±2.69e+001 $-$	9.61e+002±2.91e+001
f_{23}	5.35e+002±1.33e+000 $\approx$	5.79e+002±1.46e+002
f_{24}	2.03e+002±3.76e+000 $-$	2.00e+002±5.88e-002
f_{25}	2.52e+002±1.54e+001 $\approx$	2.86e+002±2.13e+002
$-$	6	
$+$	1	
$\approx$	18	

Table 7 presents the average and standard deviation of the best function error values achieved by DESSA-CoDE and DESSA-CoDE* on each of the 25 test function over 25 independent runs using 3000 FEs and the results of the Wilcoxon rank-sum tests conducted to compare them. It can be observed that DESSA-CoDE* performed better than DESSA-CoDE as DESSA-CoDE* outperformed DESSA-CoDE on 6 test functions while DESSA-CoDE is superior to DESSA-CoDE* on only 1 test function, thereby supporting our claim that DESSA has the ability of accommodating more strategies and parameter settings.

5. Conclusions and Discussion

The performance of DE highly depends on its trial vector generation strategy and control parameter values. In the past few years, great efforts have been made to automate the strategy selection or parameter tuning procedure of DE and quite a few DE variants have emerged, such as jDE, JADE, SaDE, EPSDE, and CoDE. Although these variants have shown certain advantages over the classical DE, they may not perform satisfactorily on CEPs.

In this paper, we have proposed to employ surrogate models to adapt the trial vector generation strategy and control parameter setting in the search process of DE for solving CEPs, and a generalized framework called DESSA has been proposed. For each target vector, DESSA generates multiple trial vectors by using different strategies and parameter settings. After that, a surrogate model is built to identify the potentially best trial vector among the generated trial vectors, which will then be evaluated with the real objective function. With this framework, three concrete DE variants, namely DESSA-CoDE, DESSA-SaDE, and DESSA-CoDE*, have been developed. Empirical results showed that DESSA-CoDE is more cost-effective than CoDE and also generally outperformed CMA-ES, SMA-EPSDE and the compared self-adaptive DE variants. By comparing DESSA-SaDE to SaDE, experimental results showed that this novel self-adaptation scheme achieved superior performance in comparison with the self-adaptation scheme employed in SaDE. Moreover,it is shown that DESSA has the ability of accommodating more search strategies by comparing DESSA-CoDE* and DESSA-CoDE. All these observations demonstrate that the new self-adaptation scheme as a more suitable self-adaptation scheme, and DESSA as a promising framework for solving CEPs.

In future work, the efficacy of the proposed self-adaptation scheme will be tested on more test functions and higher dimensions and other modeling techniques will be considered. Also, the combination of the idea of CRADE with that of DESSA will be tested on some benchmark functions at the expectation of obtaining a better DE variant for solving CEPs. Since DESSA is capable of accommodating different sets of trial vector generation strategies and control parameter settings, it is unlikely that all DE variants designed based on DESSA will perform the same. Hence, how to identify a good set of search strategies and parameter settings for DESSA would be another direction for further investigation.

Acknowledgment

This work was supported in part by the 973 Program of China under Grant 2011CB707006, the National Natural Science Foundation of China under Grants 61175065 and 61329302, the Program for New Century Excellent Talents in University under Grant NCET-12-0512, the Science and Technological Fund of Anhui Province for Outstanding Youth under Grant 1108085J16, and the European Union Seventh Framework Programme under Grant 247619.

References

[1] R. Storn, K. Price, Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, International Computer Science Institute-Publications-TR, 1995.

[2] R. Storn, On the usage of differential evolution for function optimization, in: Proceedings of the North American Fuzzy Information Processing Society (NAFIPS-1996), IEEE, 1996, pp. 519–523.

[3] K. Price, R. Storn, J. Lampinen, Differential evolution: a practical approach to global optimization, Springer-Verlag New York, Inc., 2005.

[4] S. Das, P. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31.

[5] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation 13 (2) (2009) 398–417.

[6] R. Gämperle, S. D. Müller, P. Koumoutsakos, A parameter study for differential evolution, Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation 10 (2002) 293–298.

[7] M. G. Omran, A. Salman, A. P. Engelbrecht, Self-adaptive differential evolution, in: Proceedings of Computational Intelligence and Security, Lecture Notes in Artificial Intelligence, Vol. 3801, Springer, 2005, pp. 192–199.

[8] A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Vol. 2, 2005, pp. 1785–1791.

[9] 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 Transactions on Evolutionary Computation 10 (6) (2006) 646–657.

[10] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, IEEE, 2008, pp. 1110–1116.

[11] J. Zhang, A. Sanderson, JADE:adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 945–958.

[12] Z. Yang, K. Tang, X. Yao, Scalability of generalized adaptive differential evolution for large-scale continuous optimization, Soft Computing 15 (11) (2011) 2141–2155.

[13] R. Mallipeddi, P. Suganthan, Q. Pan, M. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied Soft Computing 11 (2) (2011) 1679–1696.

[14] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 55–66.

[15] W. Gong, Á. Fialho, Z. Cai, Adaptive strategy selection in differential evolution, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation, ACM, 2010, pp. 409–416.

[16] Á. Fialho, R. Ros, M. Schoenauer, M. Sebag, Comparison-based adaptive strategy selection with bandits in differential evolution, in: Proceeding of Parallel Problem Solving from Nature–PPSN XI, Springer, 2010, pp. 194–203.

[17] P. Hajela, J. Lee, Genetic algorithms in multidisciplinary rotor blade design, in: Proceedings of the 36th Conference on Structures, Structural Dynamics, and Materials, New Orleans, LA, USA, 1995, pp. 2187–2197.

[18] Y. Jin, M. Olhofer, B. Sendhoff, Managing approximate models in evolutionary aerodynamic design optimization, in: Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Vol. 1, 2001, pp. 592–599.

[19] M. Farina, J. Sykulski, Comparative study of evolution strategies combined with approximation techniques for practical electromagnetic optimization problems, IEEE Transaction on Magnetics 37 (5) (2001) 3216–3220.

[20] Y. S. Ong, P. B. Nair, A. J. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling, AIAA journal 41 (4) (2003) 687–696.

[21] P. Zhang, X. Yao, L. Jia, B. Sendhoff, T. Schnier, Target shape design optimization by evolving splines, in: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 2009–2016.

[22] K. Deb, A. Srinivasan, Innovization: Discovery of innovative design principles through multiobjective evolutionary optimization, in: Multiobjective Problem Solving from Nature, Springer, 2008, pp. 243–262.

[23] Y. Ong, P. Nair, A. Keane, K. Wong, Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems, Studies in Fuzziness and Soft Computing 167 (2004) 307–332.

[24] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.

[25] F. Peng, K. Tang, G. Chen, X. Yao, Population-based algorithm portfolios for numerical optimization, IEEE Transactions on Evolutionary Computation 14 (5) (2010) 782–800.

[26] J. Zhang, A. Sanderson, DE-AEC: a differential evolution algorithm based on adaptive evolution control, in: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 3824–3830.

[27] J. Zhang, A. Sanderson, Surrogate model-based differential evolution, Adaptive Differential Evolution (2009) 83–93.

[28] X. Lu, K. Tang, X. Yao, Classification-assisted differential evolution for computationally expensive problems, in: Proceedings of the 2011 IEEE Congress on Evolutionary Computation, 2011, pp. 1986–1993.

[29] X. Lu, K. Tang, Classification- and regression-assisted differential evolution for computationally expensive problems, Journal of Computer Science and Technology 27 (5) (2012) 1024–1034.

[30] R. Mallipeddi, M. Lee, Surrogate model assisted ensemble differential evolution algorithm, in: Proceedings of the 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.

[31] E. Krempser, H. Bernardino, H. Barbosa, A. Lemonge, Differential evolution assisted by surrogate models for structural optimization problems, in: Proceedings of the Eighth International Conference on Engineering Computational Technology, Civil-Comp Press, Stirlingshire, UK, Paper 49, 2012. doi:10.4203/ccp.100.49.

[32] Á. Fialho, M. Schoenauer, M. Sebag, Toward comparison-based adaptive operator selection, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, ACM, 2010, pp. 767–774.

[33] A. W. Iorio, X. Li, Solving rotated multi-objective optimization problems using differential evolution, in: AI 2004: Advances in Artificial Intelligence, Springer, 2005, pp. 861–872.

[34] T. Runarsson, Ordinal regression in evolutionary computation, in: Proceedings of the Parallel Problem Solving from Nature–PPSN IX, Springer, 2006, pp. 1048–1057.

[35] I. Loshchilov, M. Schoenauer, M. Sebag, Comparison-based optimizers need comparison-based surrogates, in: Proceedings of the Parallel Problem Solving from Nature–PPSN XI, Springer, 2011, pp. 364–373.

[36] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge University Press, 2004.

[37] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, KanGAL Report 2005005.

[38] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 107–125.

[39] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary computation 9 (2) (2001) 159–195.