Adaptive Strategy Selection in Differential Evolution

Wenyin Gong, Álvaro Fialho, Zhihua Cai

Adaptive Strategy Selection in Differential Evolution

Wenyin Gong
School of Computer Science
China University of Geosciences
Wuhan, 430074 P.R. China
cug11100304@yahoo.com.cn

Álvaro Fialho
Microsoft Research – INRIA Joint Centre
Parc Orsay Université
91893 Orsay, France
alvaro.fialho@inria.fr

Zhihua Cai
School of Computer Science
China University of Geosciences
Wuhan, 430074 P.R. China
zhcai@cug.edu.cn

ABSTRACT

Differential evolution (DE) is a simple yet powerful evolutionary algorithm for global numerical optimization. Different strategies have been proposed for the offspring generation; but the selection of which of them should be applied is critical for the DE performance, besides being problem-dependent. In this paper, the probability matching technique is employed in DE to autonomously select the most suitable strategy while solving the problem. Four credit assignment methods, that update the known performance of each strategy based on the relative fitness improvement achieved by its recent applications, are analyzed. To evaluate the performance of our approach, thirteen widely used benchmark functions are used. Experimental results confirm that our approach is able to adaptively choose the suitable strategy for different problems. Compared to classical DE algorithms and to a recently proposed adaptive scheme (SaDE), it obtains better results in most of the functions, in terms of the quality of the final results and convergence speed.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence: Problem Solving, Control Methods, and Search; G.1.6 [**Numerical Analysis**]: Optimization

General Terms

Algorithms

1. INTRODUCTION

Differential evolution (DE), proposed by Storn and Price [20], is an efficient and versatile population-based, direct search algorithm for global optimization. Among DE advantages are its simple structure, ease of use, speed, and robustness, which allows its application on many real-world applications, such as data mining [1, 4], pattern recognition, digital filter design, neural network training, etc. [15, 6, 3]. Most recently, DE has also been used for the global permutation-based combinatorial problems [13].

GECCO'10, July 7–11, 2010, Portland, Oregon, USA.

In the seminal DE algorithm [20], a single mutation strategy was used. Later on, Price and Storn suggested ten other different strategies in [15, 21]. However, the selection of which of such strategies should be used is a difficult and crucial task for the performance of the DE, besides being problem-dependent [17, 16]. Some approaches have already been proposed to do this strategy selection in an autonomous way, as follows. Xie and Zhang [24] presented a swarm algorithm framework, in which a neural network is used to adaptively update the weights of the DE strategies. In [27], Zamuda *et al.* used a fixed parameter r_s to choose the strategy among three available ones. Qin *et al.* [17, 16] proposed a variant of DE, namely SaDE, that implements different strategies and also updates their weights in the search based on their previous success rate. They also used SaDE on constrained problems [9]. In [2] and [25] strategy adaptation techniques similar to SaDE are also used to enhance DE performance. To the best of our knowledge, the study on adaptive strategy selection in DE is still scarce.

In order to alleviate this drawback, in this paper, the probability matching technique is integrated into DE to implement the adaptive strategy selection for different problems. Four credit assignment methods, that reward each strategy based on the relative fitness improvement achieved by its recent applications, are analyzed.

Experiments have been conducted on 13 widely used benchmark problems. The results confirm that our approach is able to efficiently select which strategy should be mostly applied at each stage of the search, while solving the problem. Compared to what would be the choices of a naïve user (*i.e.*, a DE applying a single strategy, and a DE that uniformly selects between the available ones), and to a different adaptive scheme recently proposed, the SaDE [16], our approach was able to obtain better results in most of the functions, in terms of convergence speed and quality of the final results.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the background and related work of this paper. Section 3 describes our proposed approach in detail, followed by the experimental results and discussions in Section 4. Finally, Section 5 is devoted to conclusions and future work.

2. BACKGROUND AND RELATED WORK

2.1 Problem Formulation

Without loss of generality, in this work, we consider the following numerical optimization problem:

$$\text{Minimize} \quad f(\mathbf{x}), \qquad \mathbf{x} \in S, \tag{1}$$

where $S \subseteq \mathbb{R}^D$ is a compact set, $\mathbf{x} = [x_1, x_2, \cdots, x_D]^T$, and D is the dimension, *i.e.*, the number of decision variables. Generally, for each variable x_j, it satisfies a boundary constraint, such that:

$$L_j \leq x_j \leq U_j, j = 1, 2, \cdots, D. \tag{2}$$

2.2 Differential Evolution

DE [20] is a simple evolutionary algorithm (EA) for global numerical optimization. It creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has an equal or better fitness value. The pseudo-code of the original DE algorithm is shown in Algorithm 1, where D is the number of decision variables; NP is the population size; F is the mutation scaling factor; CR is the crossover rate; $x_{i,j}$ is the j-th variable of the solution $\mathbf{x}_i$; $\mathbf{u}_i$ is the offspring; $\mathrm{rndint}(1, D)$ is a uniformly distributed random integer number between 1 and D; and $\mathrm{rndreal}_j[0, 1)$ is a uniformly distributed random real number in $[0, 1)$, generated anew for each value of j. Many mutation strategies to create a candidate are available. In Algorithm 1, the use of the classic "DE/rand/1/bin" strategy is illustrated (see line 9).

Algorithm 1 The DE algorithm with DE/rand/1/bin strategy

```
1: Generate the initial population
2: Evaluate the fitness for each individual
3: while the halting criterion is not satisfied do
4:    for i = 1 to NP do
5:       Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
6:       j_rand = rndint(1, D)
7:       for j = 1 to D do
8:          if rndreal_j[0, 1) < CR or j is equal to j_rand then
9:             u_{i,j} = x_{r1,j} + F · (x_{r2,j} − x_{r3,j})
10:         else
11:            u_{i,j} = x_{i,j}
12:         end if
13:      end for
14:   end for
15:   for i = 1 to NP do
16:      Evaluate the offspring u_i
17:      if f(u_i) is better than or equal to f(x_i) then
18:         Replace x_i with u_i
19:      end if
20:   end for
21: end while
```

From Algorithm 1, we can see that there are only three control parameters in DE, being them NP, F and CR. As for the terminal conditions, we can either fix the maximum number of fitness function evaluations (*Max_NFFEs*) or define a desired solution value to be reached (*VTR*).

2.3 Adaptive Strategy Selection

Typically, the parameter setting in EAs is done before launching the main runs that will be used to assess the algorithm, according to the user's experience, or by an *external tuning* method. The main drawback of such *off-line* methods is that they define a static setting, what leads to sub-optimal performance. Intuitively, as the algorithm proceeds from a global (early) exploration of the landscape to a more focused, exploitation-like behavior, the parameters should be adjusted to take care of this new reality. Indeed, it has been empirically and theoretically demonstrated that different values of parameters might be optimal at different stages of the search process (see [5, p.21] and references therein).

Following [5], the *internal control* of the parameters can be done in different ways. *Deterministic* methods modify the parameters values according to predefined rules; *Self-Adaptive* methods encode the parameters within the genotype, which is thus evolved in parallel with the solution; and lastly, the *Adaptive* methods use changes in some particular properties of the search process as an input signal to modify the parameter values. While the first approach introduces the extra difficulty of defining the control rules, the second defines the parameters *for free*, but the parameters space is merged with the solutions space, thus augmenting the overall complexity of the search.

This paper is focused on the latter approach, more specifically, on the *Adaptive Strategy Selection* (AdapSS). Inspired by some recent works in the Genetic Algorithms community (see, *e.g.*, [23, 7]), its objective is to automatically select between the available (possibly ill-known) mutation strategies, according to their performance on the current search/optimization process. To do so, there is the need for two components: the credit assignment, that defines how the impact of the strategies on the search should be assessed and transformed into a numerical reward; and the strategy (or operator) selection mechanism that, based on the rewards received, select which strategy should be applied at the given moment of the search.

Algorithm 2 Probability matching-based DE with adaptive strategy selection: PM-AdapSS-DE

```
1: Set CR = 0.9, F = 0.5 and NP = 100
2: Generate the initial population
3: Evaluate the fitness for each individual
4: Set the generation counter t = 1
5: Set K = 4, p_min = 0.05, and α = 0.3                          ⇐
6: For each strategy a, set q_a(t) = 0 and p_a(t) = 1/K          ⇐
7: while The halting criterion is not satisfied do
8:    for i = 1 to NP do
9:       Select the strategy SI_i based on its probability       ⇐
10:      Select uniform randomly r1 ≠ r2 ≠ r3 ≠ r4 ≠ r5 ≠ i
11:      j_rand = rndint(1, D)
12:      for j = 1 to D do
13:         if rndreal_j[0, 1) < CR or j == j_rand then
14:            if SI_i == 1 then
15:               u_{i,j} is generated by "DE/rand/1" strategy
16:            else if SI_i == 2 then
17:               u_{i,j} is generated by "DE/rand/2" strategy
18:            else if SI_i == 3 then
19:               u_{i,j} is generated by "DE/rand-to-best/2" strategy
20:            else if SI_i == 4 then
21:               u_{i,j} is generated by "DE/current-to-rand/1"
22:            end if
23:         else
24:            u_{i,j} = x_{i,j}
25:         end if
26:      end for
27:   end for
28:   for i = 1 to NP do
29:      Evaluate the offspring u_i
30:      if f(u_i) is better than or equal to f(x_i) then
31:         Calculate η_i using Eqn. (5)                         ⇐
32:         Replace x_i with u_i
33:      else
34:         Set η_i = 0                                          ⇐
35:      end if
36:      S_{SI_i} ← η_i                                          ⇐
37:   end for
38:   Calculate the reward r_a(t) for each strategy              ⇐
39:   Update the quality q_a(t) for each strategy                ⇐
40:   Update the probability p_a(t) for each strategy            ⇐
41:   t = t + 1
42: end while
```

3. OUR APPROACH: PM-ADAPSS-DE

As previously mentioned, there are many mutation strategies in DE, each one presenting different characteristics and being suitable for different problems. However, choosing the best strategy for a problem at hand is a difficult task. In this work, we propose the utilization of *Probability Matching* (PM) [8] for the autonomous strat-

egy selection on the DE, what we refer to as PM-AdapSS-DE. The main objectives of this work are two-fold. First, the PM method is integrated into DE to implement the adaptive strategy selection. Second, four credit assignment techniques based on the relative fitness improvement are compared. Three crucial issues behind the PM-AdapSS-DE algorithm are elucidated as follows.

3.1 Strategy Selection: Probability Matching

Suppose we have $K > 1$ strategies in the pool $A = \{a_1, \cdots, a_K\}$ and a probability vector $\mathbf{P}(t) = \{p_1(t), \cdots, p_K(t)\}$ ($\forall t : p_{min} \leq p_i(t) \leq 1; \sum_{i=1}^{K} p_i(t) = 1$). In this work, the PM technique is used to adaptively update the probability $p_a(t)$ of each strategy a based on its known performance (frequently updated by the rewards received). Denote $r_a(t)$ as the reward that a strategy a receives after its application at time t. $q_a(t)$ is the known quality (or empirical estimate) of a strategy a, that is updated as follows [23]:

$$q_a(t+1) = q_a(t) + \alpha[r_a(t) - q_a(t)], \tag{3}$$

where $\alpha \in (0, 1]$ is the adaptation rate. Based on this quality estimate, the PM method updates the probability $p_a(t)$ of applying each operator as follows [8], [23]:

$$p_a(t+1) = p_{min} + (1 - K \cdot p_{min}) \frac{q_a(t+1)}{\sum_{i=1}^{K} q_i(t+1)}. \tag{4}$$

where $p_{min} \in (0, 1)$ is the minimal probability value of each strategy, used to ensure that no operator gets lost [23].

3.2 Credit Assignment

In order to assign the credit for each strategy, we adopt the relative fitness improvement η_i proposed in [12] as follows:

$$\eta_i = \frac{\delta}{cf_i} \cdot |pf_i - cf_i| \tag{5}$$

where $i = 1, \cdots, NP$. δ is the fitness of the best-so-far solution in the population. pf_i and cf_i are the fitness of the target parent and of its offspring, respectively. Note that if no improvement (*i.e.*, the offspring is worse than or equal to its target parent) is achieved, a null credit is assigned.

Denote S_a as the set of all relative fitness improvements achieved by the application of a strategy $a (a = 1, \cdots, K)$ during generation t. At the end of the generation, an unique reward is used to update the quality measure kept by the PM method (Eq. 3). Following [7], to extract such reward from S_a, we analyze four different credit assignment methods as follows:

- **AverageAbsoluteReward** (AvgAbs):

$$r_a(t) = \frac{\sum_{i=1}^{|S_a|} S_a(i)}{|S_a|} \tag{6}$$

  where $|S_a|$ is the number of elements in S_a. If $|S_a| = 0$, $r_a(t) = 0$.

- **AverageNormalizedReward** (AvgNorm):

$$r'_a(t) = \frac{\sum_{i=1}^{|S_a|} S_a(i)}{|S_a|}; \text{ and } r_a(t) = \frac{r'_a(t)}{\max\limits_{b=1,\cdots,K} r'_b(t)} \tag{7}$$

- **ExtremeAbsoluteReward** (ExtAbs):

$$r_a(t) = \max_{i=1,\cdots,|S_a|} S_a(i) \tag{8}$$

- **ExtremeNormalizedReward** (ExtNorm):

$$r'_a(t) = \max_{i=1,\cdots,|S_a|} S_a(i); \text{ and } r_a(t) = \frac{r'_a(t)}{\max\limits_{b=1,\cdots,K} r'_b(t)} \tag{9}$$

3.3 Strategy Pool

In DE, many schemes have been proposed [15, 21], applying different mutation strategies and/or recombination operations in the reproduction stage. In order to constitute the strategy pool used in this work, we have chosen four strategies that were also used in SaDE [16], listed as follows:

1) "DE/rand/1":

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) \tag{10}$$

2) "DE/rand/2":

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right) \tag{11}$$

3) "DE/rand-to-best/2":

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{best} - \mathbf{x}_{r_1}\right) + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F \cdot \left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right) \tag{12}$$

4) "DE/current-to-rand/1":

$$\mathbf{v}_i = \mathbf{x}_i + F \cdot \left(\mathbf{x}_{r_1} - \mathbf{x}_i\right) + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) \tag{13}$$

where $\mathbf{x}_{best}$ is the best individual in the current generation, r_1, r_2, $r_3, r_4, r_5 \in \{1, \cdots, NP\}$, and $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$. F is the scaling factor. NP is the population size.

All of them are controlled by the DE crossover rate CR. Note that other strategies could also be introduced in the pool; these four strategies can be seen as instances used as test-bed for the evaluation of the proposed method.

3.4 DE with Adaptive Strategy Selection

By combining the above-mentioned three aspects with the DE algorithm, the PM-AdapSS-DE method is developed. The pseudocode of PM-AdapSS-DE is illustrated in Algorithm 2. Modified steps with respect to the classical DE algorithm are marked with a left arrow "⇐". At each generation t, for each target parent i, a strategy SI_i is selected based on the probability of each strategy. Then the offspring is generated by the application of the selected strategy. After evaluating the offspring, the relative fitness improvement η_i is calculated and stored in the set S_{SI_i}. Consequently, the reward, quality, and probability of each strategy are updated at the end of each generation.

Note that in SaDE [16], the strategy adaptation is also implemented. However, our approach is completely different from SaDE in the credit assignment. In SaDE, the success and failure number of runs are considered, while the relative fitness improvement is used in our approach.

4. EXPERIMENTAL ANALYSIS

In order to evaluate the performance of our approach, 13 benchmark functions ($f_{01} - f_{13}$) were selected from [26] as the test suit. Functions $f_{01} - f_{04}$ are unimodal. The Rosenbrock's function f_{05} is a multi-modal function when $D > 3$ [18]. Function f_{06} is the step function, which has one minimum and is discontinuous. Function f_{07} is a noisy quartic function. Functions $f_{08} - f_{13}$ are multimodal functions where the number of local minima increases exponentially with the problem dimension. They appear to be the most difficult class of problems for many optimization algorithms. Due to the space limitation, we omit their descriptions here, more details can be found in [26].

Table 1: Comparison on the Error values of different credit assignment methods for all functions at $D = 30$.

F	Uniform-DE	SaDE	PM-AdapSS-DE-1	PM-AdapSS-DE-2	PM-AdapSS-DE-3	PM-AdapSS-DE-4
f_{01}	2.35E-32 ± 1.45E-32†	7.64E-41 ± 1.02E-40†	**3.38E-48 ± 5.37E-48**	**3.08E-48 ± 3.24E-48**	1.13E-45 ± 1.26E-45†	3.66E-45 ± 5.06E-45†
f_{02}	3.10E-21 ± 1.22E-21†	1.28E-26 ± 6.22E-27†	**3.57E-31 ± 6.30E-31**	**8.41E-32 ± 5.14E-32**	5.71E-29 ± 1.01E-28†	1.28E-29 ± 6.24E-30†
f_{03}	6.69E-27 ± 2.07E-26†	3.93E-32 ± 8.30E-32†	3.84E-36 ± 9.37E-36†	7.47E-36 ± 3.08E-35†	**2.12E-37 ± 4.39E-37**	**7.11E-37 ± 2.70E-36**
f_{04}	**3.44E-10 ± 1.24E-09**†	2.55E-06 ± 5.34E-06†	3.17E-09 ± 9.70E-09†	8.50E-08 ± 4.42E-07†	**7.37E-11 ± 2.95E-10**	3.53E-10 ± 1.20E-09
f_{05}	**1.24E-29 ± 2.79E-29**	2.39E-01 ± 9.56E-01	2.39E-01 ± 9.56E-01	**7.97E-02 ± 5.64E-01**	1.59E-01 ± 7.89E-01	2.39E-01 ± 9.56E-01
f_{06}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
f_{07}	1.68E-03 ± 4.16E-04†	1.52E-03 ± 4.63E-04†	9.78E-04 ± 3.21E-04	**9.61E-04 ± 2.68E-04**	9.82E-04 ± 3.69E-04	**9.63E-04 ± 2.93E-04**
f_{08}	7.42E+03 ± 2.51E+02†	7.36E+03 ± 2.26E+02†	7.28E+03 ± 2.53E+02	**7.18E+03 ± 3.05E+02**	**7.18E+03 ± 3.56E+02**	7.19E+03 ± 3.25E+02
f_{09}	1.53E+02 ± 9.73E+00†	1.51E+02 ± 8.90E+00†	1.40E+02 ± 1.09E+01	1.42E+02 ± 9.79E+00†	**1.38E+02 ± 1.02E+01**	**1.40E+02 ± 9.33E+00**
f_{10}	4.14E-15 ± 0.00E+00	4.14E-15 ± 0.00E+00	4.14E-15 ± 0.00E+00	**4.07E-15 ± 5.02E-16**	4.14E-15 ± 0.00E+00	4.14E-15 ± 0.00E+00
f_{11}	1.97E-04 ± 1.39E-03	1.97E-04 ± 1.39E-03	3.45E-04 ± 1.73E-03	3.94E-04 ± 1.95E-03	**1.48E-04 ± 1.05E-03**	**0.00E+00 ± 0.00E+00**
f_{12}	1.59E-32 ± 9.04E-34	**1.57E-32 ± 0.00E+00**	**1.57E-32 ± 0.00E+00**	**1.57E-32 ± 0.00E+00**	**1.57E-32 ± 0.00E+00**	**1.57E-32 ± 0.00E+00**
f_{13}	1.35E-30 ± 7.87E-30†	**1.35E-32 ± 0.00E+00**	**1.35E-32 ± 0.00E+00**	**1.35E-32 ± 0.00E+00**	**1.35E-32 ± 0.00E+00**	**1.35E-32 ± 0.00E+00**

† indicates the best algorithm is significantly better than its competitor by the Wilcoxon signed-rank test at $\alpha = 0.05$.

Table 2: Comparison on the NFFEs of different credit assignment methods for all functions at $D = 30$. Since all of the algorithms obtain similar S_r values, we omit to report them in this table.

F	Uniform-DE	SaDE	PM-AdapSS-DE-1	PM-AdapSS-DE-2	PM-AdapSS-DE-3	PM-AdapSS-DE-4
f_{01}	5.18E+04 ± 8.46E+02	4.28E+04 ± 7.31E+02	**3.57E+04± 7.92E+02**	**3.57E+04 ± 6.63E+02**	3.77E+04 ± 6.78E+02	3.80E+04 ± 6.79E+02
f_{02}	8.96E+04 ± 1.06E+03	7.35E+04 ± 9.73E+02	**6.18E+04± 4.31E+03**	**6.06E+04 ± 1.03E+03**	6.56E+04 ± 5.14E+03	6.47E+04 ± 8.52E+02
f_{03}	1.97E+05 ± 7.28E+03	1.65E+05 ± 8.13E+03	1.46E+05 ± 6.02E+03	1.47E+05 ± 6.01E+03	**1.44E+05± 7.09E+03**	**1.43E+05 ± 8.14E+03**
f_{04}	**3.52E+05 ± 6.93E+04**	4.29E+05 ± 3.18E+04	3.94E+05 ± 4.42E+04	4.25E+05 ± 4.50E+04	**3.31E+05 ± 5.68E+04**	3.59E+05 ± 5.47E+04
f_{05}	2.32E+05 ± 7.88E+03	2.29E+05 ± 8.66E+03	**2.00E+05± 6.60E+03**	**1.99E+05 ± 6.43E+03**	2.01E+05 ± 8.16E+03	2.02E+05 ± 7.73E+03
f_{06}	1.91E+04 ± 5.50E+02	1.63E+04 ± 4.68E+02	**1.28E+04 ± 4.93E+02**	**1.28E+04± 5.53E+02**	1.35E+04 ± 5.83E+02	1.37E+04 ± 6.30E+02
f_{07}	5.39E+04 ± 1.45E+04	4.52E+04 ± 1.05E+04	3.04E+04 ± 8.26E+03	2.92E+04 ± 7.73E+03	**2.59E+04 ± 6.52E+03**	**2.67E+04± 6.64E+03**
f_{10}	8.09E+04 ± 7.85E+02	6.60E+04 ± 9.39E+02	**5.56E+04 ± 9.26E+02**	**5.58E+04± 9.66E+02**	5.98E+04 ± 6.66E+02	6.00E+04 ± 1.10E+03
f_{11}	5.34E+04 ± 9.05E+02	4.45E+04 ± 9.16E+02	**3.72E+04 ± 7.86E+02**	**3.74E+04± 1.40E+03**	3.94E+04 ± 9.00E+02	3.96E+04 ± 8.21E+02
f_{12}	4.68E+04 ± 8.64E+02	3.86E+04 ± 7.59E+02	**3.12E+04 ± 1.22E+03**	**3.14E+04± 7.79E+02**	3.17E+04 ± 9.16E+02	3.26E+04 ± 7.86E+02
f_{13}	5.58E+04 ± 1.87E+03	4.60E+04 ± 1.14E+03	**3.81E+04 ± 1.12E+03**	**3.84E+04± 1.31E+03**	3.87E+04 ± 1.50E+03	3.96E+04 ± 1.26E+03

4.1 Experimental Settings

In the first set of experiments, presented in Section 4.3, we compare the performance of PM-AdapSS-DE implementing each of the different credit assignment methods described in Section 3.2. Therefore, there are four PM-AdapSS-DE variants: PM-AdapSS-DE-1 with *AvgAbs*, PM-AdapSS-DE-2 with *AvgNorm*, PM-AdapSS-DE-3 with *ExtAbs*, and PM-AdapSS-DE-4 with *ExtNorm*. In addition, a DE algorithm with the uniform strategy selection (Uniform-DE) is also implemented as baseline: for the creation of each offspring, a strategy is uniformly drawn from the pool. Our approach is also compared with SaDE [16]; but since the objective here is to assess their performance w.r.t. adaptive strategy selection under the same conditions, the SaDE is applied with fixed CR and F, and its fourth strategy is also controlled by CR.

In the second set of experiments, presented in Section 4.4, the performance gain obtained by the adaptive selection of strategies is analyzed by comparing the PM-AdapSS-DE with the classical DE algorithm, that implements just one of the strategies.

For all experiments, we use the following parameters unless a change is mentioned.

- Dimension of each function: $D = 30$;
- Population size: $NP = 100$, Crossover rate: $CR = 0.9$, Mutation scaling factor $F = 0.5$;
- Number of strategies: $K = 4$; PM minimal probability: $p_{min} = 0.05$, and adaptation rate: $\alpha = 0.3$;
- Value to reach (VTR): For functions $f_{01} - f_{06}$ and $f_{08} - f_{13}$, VTR = 10^{-8}; for functions f_{07}, VTR = 10^{-2};
- Max_NFFEs[1]: For f_{01}, f_{06}, f_{10}, f_{12}, and f_{13}, Max_NFFEs = $150,000$; for $f_{03} - f_{05}$, Max_NFFEs = $500,000$; for f_{02} and f_{11}, Max_NFFEs = $200,000$; for $f_{07} - f_{09}$, Max_NFFEs = $300,000$.

[1]The Max_NFFEs for all functions are mainly set as in [26], except for f_{05}, f_{08}, and f_{09}, they are less than the values used in [26].

Moreover, in our experiments, each function is optimized over 50 independent runs. To avoid any initialization bias, we also use the same set of initial random populations to evaluate the different algorithms, as done in [11].

4.2 Performance Criteria

Four performance criteria were selected from the literature [22] to evaluate the performance of the algorithms. These criteria are described as follows.

- **Error**: The error of a solution $\mathbf{x}$ is defined as $f(\mathbf{x}) - f(\mathbf{x}^*)$, where $\mathbf{x}^*$ is the global minimum of the function. The minimum error is recorded when the Max_NFFEs is reached in 50 runs. The average and standard deviation of the error values are calculated as well.
- **NFFEs**: The NFFEs is also recorded when the VTR is reached. The average and standard deviation of the NFFEs values are calculated.
- **Successful rate** (S_r): A run is considered successful if the algorithm is able to reach a function value no worse than the VTR before the Max_NFFEs condition terminates the trial. The successful rate S_r is calculated as the number of successful runs divided by the total number of runs.
- **Convergence graphs**: The convergence graphs show the median error performance of the best solution over the total runs, in the respective experiments.

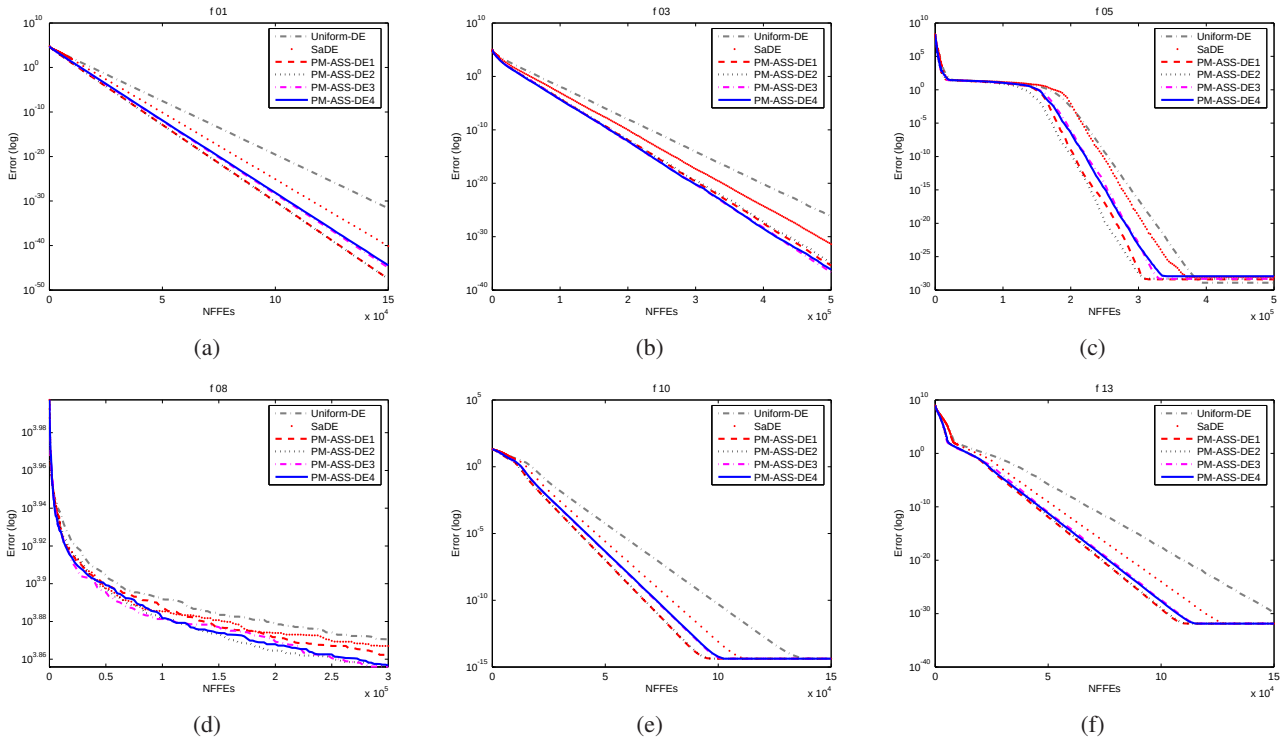

Figure 1: Convergence graph of Uniform-DE, SaDE, PM-AdapSS-DE-1, PM-AdapSS-DE-2, PM-AdapSS-DE-3, and PM-AdapSS-DE-4 on the selected functions. (a) f_{01}. (b) f_{03}. (c) f_{05}. (d) f_{08}. (e) f_{10}. (f) f_{13}.

4.3 Comparison on Different Credit Assignment Methods

In this section, we compare the performance of different credit assignment methods proposed in Section 3.2. In addition, Uniform-DE and SaDE are also compared. The results are shown in Tables 1 and 2. All results are averaged over 50 independent runs. The best and the second best results are highlighted, respectively, in **grey boldface** and **boldface**. In Table 1, the paired Wilcoxon signed-rank test at $\alpha = 0.05$ is adopted to compare the significance between two algorithms. The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test, which can be used as an alternative to the paired t-test when the results cannot be assumed to be normally distributed [19]. Additionally, some representative convergence graphs are plotted in Figure 1.

With respect to the quality of the final results, Table 1 indicates that our proposed PM-AdapSS-DE (with different credit assignment methods) is able to obtain better results than Uniform-DE and SaDE on most of the test functions. Exceptionally, on function f_{05}, Uniform-DE provides the best results; however, there is no significant difference w.r.t. the other five algorithms.

Considering the convergence speed, Table 2 and Figure 1 indicate that PM-AdapSS-DE consistently converges faster than Uniform-DE and SaDE on most of the functions.

In general, our proposed PM-AdapSS-DE approach obtains better results than Uniform-DE and SaDE in terms of the error values and the convergence rate, which might reflect that the relative fitness improvement based credit assignment techniques are better than the method used in SaDE (based on the frequency of fitness improvements). In addition, from Tables 1 and 2, it can be seen that PM-AdapSS-DE with the averaged reward (PM-AdapSS-DE-1 and PM-AdapSS-DE-2) is better than PM-AdapSS-DE with the extreme reward (PM-AdapSS-DE-3 and PM-AdapSS-DE-4) in most of the functions. Both PM-AdapSS-DE-1 and PM-AdapSS-DE-2 provide the most competitive results. Thus, in the following section, we only analyze the adaptation characteristics of PM-AdapSS-DE-1 (referred to as PM-AdapSS-DE for the sake of clearness).

4.4 Analysis of Strategy Adaptation

In this section, we compare the performance of PM-AdapSS-DE with the classical DE algorithm. In the classical DE algorithm, each single strategy in the pool used in this work is implemented and compared with PM-AdapSS-DE, in order to analyze the adaptation characteristics of our approach. For the classical DE algorithm, the parameter settings are kept the same as described in Section 4.1. The results are shown in Tables 3 and 4. Some representative convergence graphs are shown in Figure 2. The evolution trend of the probability of each strategy is plotted in Figure 3. In the last row of Table 3, according to the Wilcoxon's test, the results are summarized as "$w/t/l$", which means that PM-AdapSS-DE wins in w functions, ties in t functions, and loses in l functions, compared with its competitors. In Tables 3 and 4, "Strategy1" means DE with "DE/rand/1/bin" strategy; "Strategy2" means DE with "DE/rand/2/bin" strategy; "Strategy3" means DE with "DE/rand-to-best/2/bin" strategy; and "Strategy4" means DE with "DE/current-to-rand/1/bin" strategy.

4.4.1 Analysis of the general performance

From Table 3, the results clearly indicate that PM-AdapSS-DE is significantly better than DE with each single strategy in the pool on most of the functions. It is better than DE with the first, second,

Table 3: Comparison on the Error values between the classical DE with single strategy and PM-AdapSS-DE for all functions at $D = 30$.

F	Strategy1	Strategy2	Strategy3	Strategy4	PM-AdapSS-DE
f_{01}	4.77E-14 ± 3.84E-14†	1.38E+02 ± 3.83E+01†	**2.45E-25 ± 1.84E-25**†	2.16E+00 ± 2.43E+00†	**3.38E-48 ± 5.37E-48**
f_{02}	4.19E-10 ± 1.95E-10†	1.50E+01 ± 4.04E+00†	**2.54E-16 ± 1.16E-16**†	2.90E-02 ± 2.28E-02†	**3.57E-31 ± 6.30E-31**
f_{03}	2.59E-11 ± 3.18E-11†	1.46E+03 ± 4.10E+02†	**1.24E-19 ± 2.22E-19**†	2.86E+01 ± 2.01E+01†	**3.84E-36 ± 9.37E-36**
f_{04}	6.47E-02 ± 1.78E-01†	6.77E+00 ± 9.29E-01†	**3.09E-20 ± 2.99E-20**‡	3.12E+00 ± 1.38E+00†	**3.17E-09 ± 9.70E-09**
f_{05}	**1.14E-11 ± 7.18E-11**†	2.93E+01 ± 1.64E+00†	**7.97E-02 ± 5.64E-01**	9.17E+01 ± 6.20E+01†	2.39E-01 ± 9.56E-01
f_{06}	**0.00E+00 ± 0.00E+00**	1.42E+02 ± 3.52E+01†	**0.00E+00 ± 0.00E+00**	3.56E+00 ± 3.67E+00†	**0.00E+00 ± 0.00E+00**
f_{07}	4.89E-03 ± 1.27E-03†	7.03E-02 ± 1.86E-02†	2.83E-03 ± 8.25E-04†	**9.21E-04 ± 3.25E-04**	**9.78E-04 ± 3.21E-04**
f_{08}	**6.61E+03 ± 6.54E+02**‡	7.34E+03 ± 2.65E+02	7.45E+03 ± 2.35E+02†	7.85E+03 ± 2.69E+02†	**7.28E+03 ± 2.53E+02**
f_{09}	**1.32E+02 ± 2.46E+01**‡	2.21E+02 ± 1.04E+01†	1.67E+02 ± 1.03E+01†	**1.31E+02 ± 7.63E+00**‡	1.40E+02 ± 1.09E+01
f_{10}	7.35E-08 ± 3.18E-08†	4.55E+00 ± 3.79E-01†	**1.70E-13 ± 6.46E-14**†	3.94E-01 ± 3.59E-01†	**4.14E-15 ± 0.00E+00**
f_{11}	**0.00E+00 ± 0.00E+00**	1.22E+00 ± 7.31E-02†	1.77E-03 ± 4.31E-03†	5.14E-01 ± 3.46E-01†	**3.45E-04 ± 1.73E-03**
f_{12}	5.07E-15 ± 6.72E-15†	3.37E+01 ± 3.94E+01†	**6.77E-26 ± 6.48E-26**†	1.57E-03 ± 3.60E-03†	**1.57E-32 ± 0.00E+00**
f_{13}	7.40E-13 ± 8.87E-13†	5.49E+03 ± 6.55E+03†	**6.43E-23 ± 9.54E-23**†	3.32E-02 ± 1.62E-01†	**1.35E-32 ± 0.00E+00**
$w/t/l$	9/2/2	12/1/0	10/2/1	11/1/1	–

† indicates PM-AdapSS-DE is significantly better than its competitor by the Wilcoxon signed-rank test at $\alpha = 0.05$.
‡ means that the corresponding algorithm is significantly better than our proposed PM-AdapSS-DE method.

Table 4: Comparison on the NFFEs between the classical DE algorithm with single strategy and PM-AdapSS-DE for all functions at $D = 30$. For each function, the successful rate S_r is shown in parenthesis.

F	Strategy1	Strategy2	Strategy3	Strategy4	PM-AdapSS-DE
f_{01}	1.05E+05 ± 2.67E+03 (1.00)	NA ± NA (0.00)	**6.44E+04 ± 1.05E+03** (1.00)	NA ± NA (0.00)	**3.57E+04 ± 7.92E+02** (1.00)
f_{02}	1.76E+05 ± 3.38E+03 (1.00)	NA ± NA (0.00)	**1.15E+05 ± 1.76E+03** (1.00)	NA ± NA (0.00)	**6.18E+04 ± 4.31E+03** (1.00)
f_{03}	4.06E+05 ± 1.66E+04 (1.00)	NA ± NA (0.00)	**2.65E+05 ± 6.23E+03** (1.00)	NA ± NA (0.00)	**1.46E+05 ± 6.02E+03** (1.00)
f_{04}	**3.36E+05 ± 5.61E+03** (0.06)	NA ± NA (0.00)	**2.29E+05 ± 5.35E+03** (1.00)	NA ± NA (0.00)	3.94E+05 ± 4.42E+04 (0.92)
f_{05}	4.35E+05 ± 1.49E+04 (1.00)	NA ± NA (0.00)	**1.95E+05 ± 5.52E+03** (0.98)	NA ± NA (0.00)	**2.00E+05 ± 6.60E+03** (0.94)
f_{06}	3.95E+04 ± 1.88E+03 (1.00)	NA ± NA (0.00)	2.51E+04 ± 9.51E+02 (1.00)	**1.10E+04 ± 1.24E+03** (0.14)	**1.28E+04 ± 4.93E+02** (1.00)
f_{07}	1.44E+05 ± 4.10E+04 (1.00)	NA ± NA (0.00)	8.82E+04 ± 2.64E+04 (1.00)	**1.91E+04 ± 5.01E+03** (1.00)	**3.04E+04 ± 8.26E+03** (1.00)
f_{10}	NA ± NA (0.00)	NA ± NA (0.00)	**1.01E+05 ± 1.60E+03** (1.00)	NA ± NA (0.00)	**5.56E+04 ± 9.26E+02** (1.00)
f_{11}	1.09E+05 ± 2.80E+03 (1.00)	NA ± NA (0.00)	**6.92E+04 ± 5.38E+03** (0.84)	NA ± NA (0.00)	**3.72E+04 ± 7.86E+02** (0.96)
f_{12}	9.59E+04 ± 2.94E+03 (1.00)	NA ± NA (0.00)	**6.15E+04 ± 1.31E+03** (1.00)	NA ± NA (0.00)	**3.12E+04 ± 1.22E+03** (1.00)
f_{13}	1.14E+05 ± 4.03E+03 (1.00)	NA ± NA (0.00)	**7.64E+04 ± 1.92E+03** (1.00)	NA ± NA (0.00)	**3.81E+04 ± 1.12E+03** (1.00)
$\sum S_r$	9.06	0.00	10.82	1.14	10.82

third, and fourth strategy on 9, 12, 10, and 11 out of 13 functions, respectively.

With respect to the convergence speed, Table 4 and Figure 2 show that on most of the test functions PM-AdapSS-DE is capable of producing the fastest convergence speed compared with the four DE algorithms.

4.4.2 Analysis of the adaptation characteristics

In order to analyze the adaptation characteristics of PM-AdapSS-DE, the evolution trend of the probability of each strategy is shown in Figure 3. According to the results shown in Tables 3, 4 and Figures 2, 3, we can observe that:

- Compared the results of DE with each single strategy, the strategy 3 obtains the overall best results, followed by the strategy 1. The results of the strategy 2 are the worst due to the high diversity produced by it. By carefully looking at the convergence speed in Figure 2, it can be seen that on all functions the strategy 4 converges faster at the beginning of the evolution. However, it may cause stagnation rapidly. The reason might be that the fourth strategy is based on the current solution and performs local search around it [14].

- From Figure 3, we can see that on most of the functions, the strategy 4 obtains the greatest probability, followed by the third, the first, and the second one. This phenomenon is reasonable. Since the strategy 4 does local search around the target solution, it is able to produce the highest relative fitness improvement and, hence, obtains the greatest probability. On the other hand, the strategy 3 is capable of providing the best results individually, so that in PM-AdapSS-DE it can generate promising offspring and obtains a greater probability. As the strategy 2 performs the worst, it gets the lowest probability on most of the functions.

- According to the results in Tables 3, 4 and Figure 2, we can see that on the majority of the functions PM-AdapSS-DE obtains the best results compared with the single strategy based DE. This is because of the cooperation among the different strategies in PM-AdapSS-DE. For example, the fourth strategy performs the local search around each target parent and converges faster; meanwhile, the third strategy generates more promising solutions and prevents the stagnation caused by the fourth strategy.

In summary, from the above analysis we can conclude that our approach is able to efficiently select the suitable strategy for different problems. For each function, one of the strategies was empirically found to be the best. The PM-AdapSS-DE was able to automatically select between them, without any external *a priori* knowledge, thus enhancing the performance of the classical DE.

5. CONCLUSIONS AND FUTURE WORK

In this paper, the probability matching method and the relative fitness improvement based credit assignment techniques are integrated into the classical DE algorithm. By implementing the Adaptive Strategy Selection paradigm, it is capable of efficiently adapting to the characteristics of the region of the landscape that is currently being explored by the algorithm, by efficiently selecting between the strategies while solving the problem.

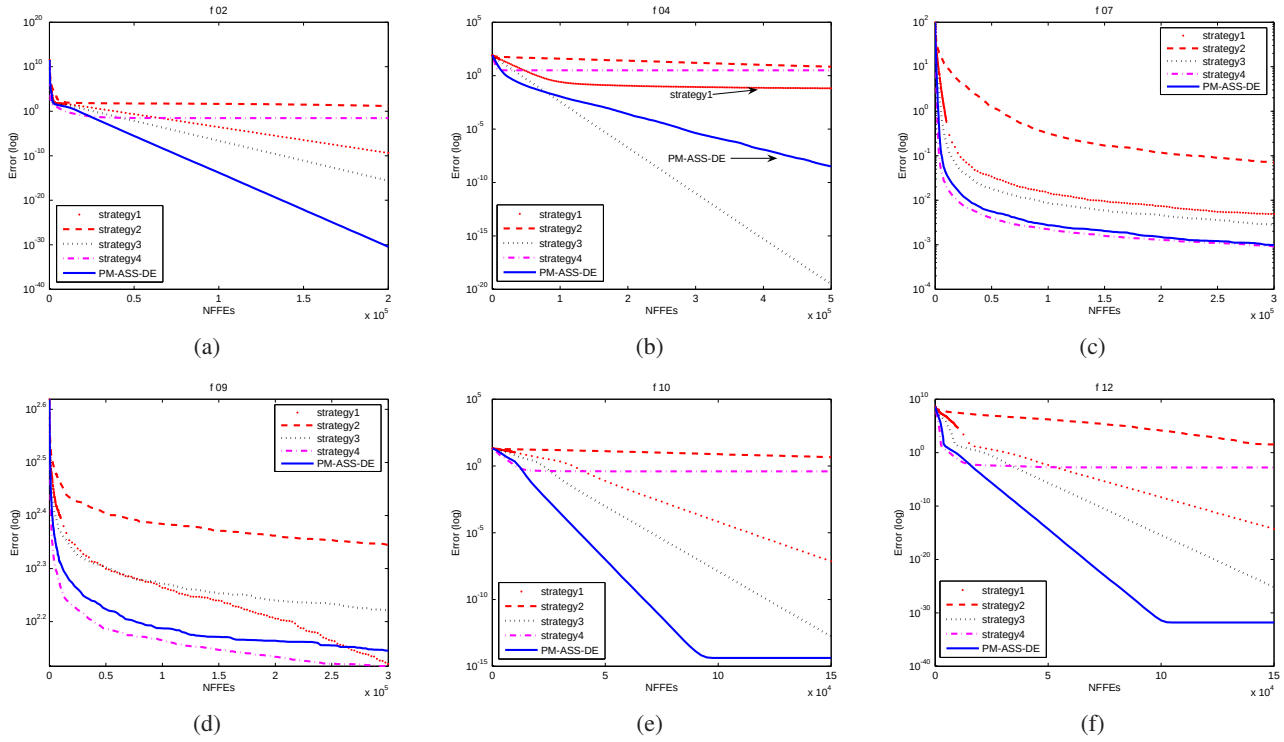

Figure 2: Convergence graph of DE with single strategy and PM-AdapSS-DE on the selected functions. (a) f_{02}. (b) f_{04}. (c) f_{07}. (d) f_{09}. (e) f_{10}. (f) f_{12}.

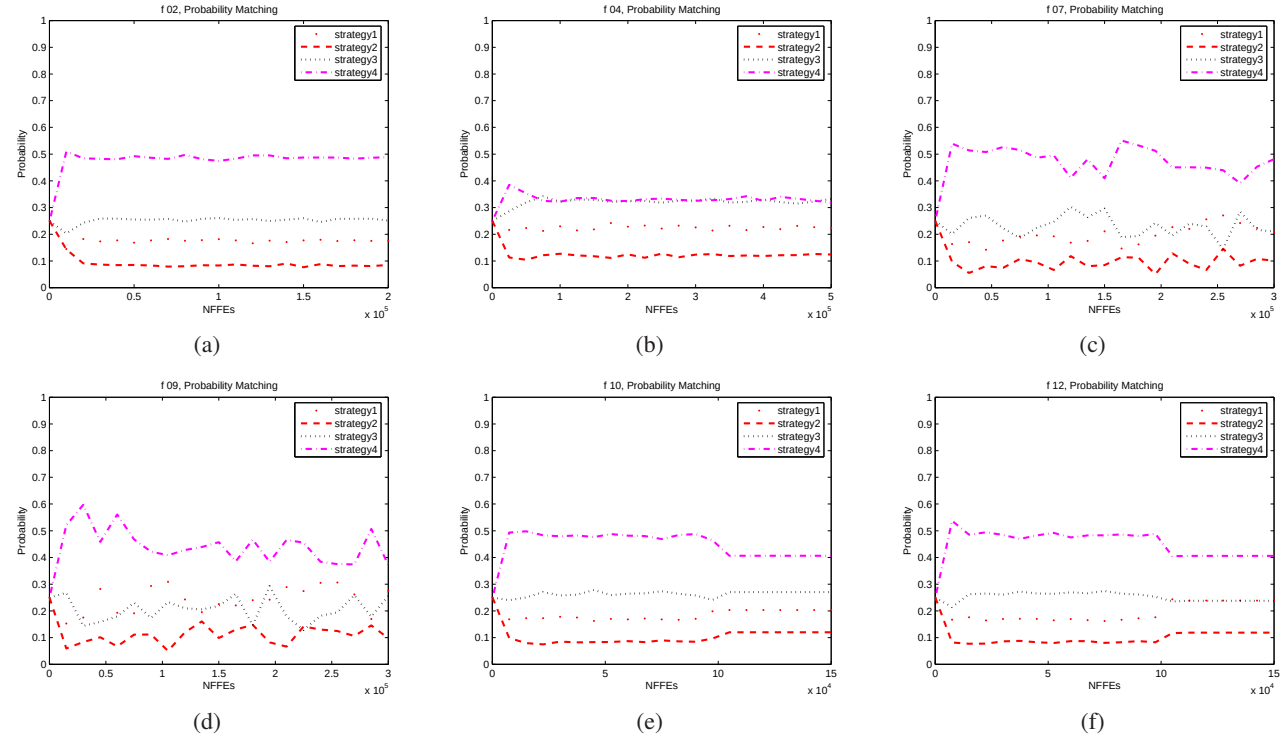

Figure 3: Adaptation characteristics of the probability of each strategy on the selected functions. (a) f_{02}. (b) f_{04}. (c) f_{07}. (d) f_{09}. (e) f_{10}. (f) f_{12}.

In our proposed PM-AdapSS-DE algorithm, four different credit assignment techniques are analyzed in combination with the Probability Matching selection rule. Their performances are compared with Uniform-DE and SaDE. In addition, PM-AdapSS-DE is compared with the classical DE algorithm using each of the strategies alone. The results indicate that PM-AdapSS-DE obtains better results in terms of the quality of the final solutions and convergence speed.

Besides the Probability Matching for strategy selection, the Adaptive Pursuit [23] and the Multi-Armed Bandit [7] approaches are also used for this purpose in the EA literature, achieving better results than PM in the analyzed cases. In the future work, we shall also analyze these techniques in the context of adaptive strategy selection within DE.

Another important issue that should be further analyzed is the credit assignment. The methods used in this work consider just the relative fitness improvement. When tackling multimodal problems, the maintenance of a minimal level of diversity is also important for the search process, and thus should also be taken into account for the rewarding of the strategies. The Compass or the Pareto-based approaches, proposed in [10], could be used.

Lastly, the on-line adaptation of CR and F was already shown to be beneficial for DE (*e.g.*, in the SaDE method), and should also be considered in the near future. In addition, the sensitivity of the two parameters p_{min} and α should also be studied in the future.

Acknowledgments

This work was partly supported by the Fund for Outstanding Doctoral Dissertation of CUG, the National High Technology Research and Development Program of China under Grant No. 2009AA12Z117, and the Research Fund for the Doctoral Program of Higher Education under Grant No. 20090145110007. The authors would like to thank the anonymous reviewers for their constructive suggestions.

6. REFERENCES

[1] B. Alatas, E. Akin, and A. Karci. MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules. *Appl. Soft Comput.*, 8(1):646–656, Jan. 2008.

[2] J. Brest, B. Boscovic, S. Greiner, V. Zumer, and M. S. Maucec. Performance comparison of adaptive and self-adaptive differential evolution algorithms. *Soft Comput.*, 11(7):617–629, May 2007.

[3] U. Chakraborty. *Advances in Differential Evolution*. Springer-Verlag, Berlin, 2008.

[4] S. Das, A. Abraham, and A. Konar. Automatic clustering using an improved differential evolution algorithm. *IEEE Trans. on Syst. Man, Cybern. A, Syst., Humans*, 38(1):218–237, Feb 2008.

[5] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In Lobo, F.G. et al., editor, *Parameter Setting in Evolutionary Algorithms*, pages 19–46. Springer, 2007.

[6] V. Feoktistov. *Differential Evolution: In Search of Solutions*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[7] Á. Fialho, M. Schoenauer, and M. Sebag. Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In *Proc. Genetic Evol. Comput. Conf.*, pages 779–786, 2009.

[8] D. E. Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Mach. Learn.*, 5(4):407–425, 1990.

[9] V. L. Huang, A. K. Qin, and P. N. Suganthan. Self-adaptive differential evolution algorithm for constrained real-parameter optimization. In *Proc. IEEE Congress on Evol. Comput.*, pages 17–24, 2006.

[10] J. Maturana, F. Lardeux, and F. Saubion. Controlling behavioral and structural parameters in evolutionary algorithms. In *Proc. EA'09*, 2009.

[11] N. Noman and H. Iba. Accelerating differential evolution using an adaptive local search. *IEEE Trans. on Evol. Comput.*, 12(1):107–125, Feb 2008.

[12] Y.-S. Ong and A. J. Keane. Meta-Lamarckian learning in memetic algorithms. *IEEE Trans. on Evol. Comput.*, 8(2):99–110, Apr 2004.

[13] G. C. Onwubolu and D. Davendra. *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*. Springer-Verlag, Berlin, 2009.

[14] K. Price and J. Ronkkonen. Comparing the uni-modal scaling performance of global and local selection in a mutation-only differential evolution algorithm. In *IEEE Congr. on Evol. Comput.*, pages 2034–2041, 2006.

[15] K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin, 2005.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. on Evol. Comput.*, 13(2):398–417, Apr 2009.

[17] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *IEEE Congr. on Evol. Comput.*, pages 1785–1791, 2005.

[18] Y.-W. Shang and Y.-H. Qiu. A note on the extended rosenbrock function. *Evol. Comput.*, 14(1):119–126, 2006.

[19] C. Shaw, K. Williams, and R. Assassa. Patients' views of a new nurse-led continence service. *Journal of Clinical Nursing*, 9(4):574–584, 2003.

[20] R. Storn and K. Price. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optim.*, 11(4):341–359, Dec 1997.

[21] R. Storn and K. Price. Home page of differential evolution, 2008.

[22] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, 2005.

[23] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proc. Genetic Evol. Comput. Conf.*, pages 1539–1546, 2005.

[24] X.-F. Xie and W.-J. Zhang. SWAF: Swarm algorithm framework for numerical optimization. In *Proc. Genetic Evol. Comput. Conf., Part I*, volume 3102 of *LNCS*, pages 238–250, 2004.

[25] Z. Yang, K. Tang, and X. Yao. Self-adaptive differential evolution with neighborhood search. In *Proc. IEEE Congress on Evol. Comput.*, pages 1110–1116, 2008.

[26] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Trans. on Evol. Comput.*, 3(2):82–102, Jul 1999.

[27] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In *2008 IEEE World Congr. on Comput. Intell.*, pages 3719–3726, 2008.