Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Differential Evolution with Polymorphic Schemes

Christian Veenhuis
Berlin University of Technology
Berlin, Germany
veenhuis@googlemail.com

Mario Köppen
Kyushu Institute of Technology
680-4, Kawazu, Iizuka, Fukuoka 820-8502, Japan
mkoeppen@pluto.ai.kyutech.ac.jp

***Abstract*—In recent years a new evolutionary algorithm for optimization in continuos spaces called Differential Evolution (DE) has developed. If one wants to apply DE one has to specify several parameters as well as to select a scheme. Several schemes being widely used can be found in literature. This raises the question, which one fits best to your application at hand. To get rid of this scheme selection problem, a new concept called Polymorphic Differential Evolution (PolyDE) is proposed. PolyDE generalizes the standard schemes by a polymorphic scheme. The mathematical expression of this polymorphic scheme can be changed on symbolic level. This polymorphic scheme is an adaptive scheme changing symbols based on accumulative histograms and roulette-wheel sampling. PolyDE is applied to four typical benchmark functions known from literature and its performance is ranked between the top and middle region compared to all standard DE schemes. Since PolyDE performs not worse than the other schemes it can be used as alternative to them solving this way the scheme selection problem. The best performance is obtained for the multimodal functions.**

***Index Terms*—Differential Evolution, Adaptive Scheme, Polymorphic Scheme, Polymorphic Equation, Polymorphic Symbol, Scheme Selection**

I. Introduction

In 1995 a new evolutionary algorithm for optimization called Differential Evolution (DE) was introduced by Storn and Price in [4]. The main difference to other evolutionary algorithms is that the mutation process is guided by other population members and not purely random. As stated in [6] DE is a fast optimizer in terms of 'needed number of evaluation steps' to minimize a function.

If one wants to apply DE one has to specify several parameters as well as to select a scheme. Specifying the parameters is not that difficult since there are good defaults proposed in [6] although other results revealed that the strategy parameters are sensitive to the application at hand [2]. To solve this, some authors [1], [3], [7] introduced adaptive DE variants, which determine the parameters by themselfs so they don't need to be pre-defined. But only few work is done concerning the scheme selection process. In literature 5 typical schemes can be found being widely used. But, which one of them fits best to your application? All your experiments need to be conducted with each of these schemes to be sure.

In [3] the authors introduced a self-adaptive DE (SADE) approach, which does not only adapt the strategy parameters. It also automatically selects one out of two schemes based on a probability p. Every 50 iterations this probability is updated based on success and failure rates of the selected schemes realizing this way an adaptive scheme selection process. Although they reported good results, this approach has two restrictions: (a) only 2 schemes are used and (b) schemes as a whole are selected, i.e., they are the 'atomic units' in this process. But it is not known in advance whether one of the pre-defined schemes fits to your problem or if a variation or new scheme is needed.

Polymorphism is well-known in nature and refers to the situation that organisms of a species can have more than one phenotype. Computer viruses using polymorphic code to fight against the pattern analysation performed by anti-virus software are another example. These viruses change their code while maintaining the same functionality. In this paper polymorphism is transferred to algorithms, especially the Differential Evolution optimiser. If the class of DE algorithms is considered as a species, then different used schemes lead to different DE instances, that is phenotypes, of this algorithm.

The main aim of the proposed concept called Polymorphic Differential Evolution (PolyDE) is to get rid of the scheme selection process. For this, an adaptive scheme with polymorphic capabilities is defined. This polymorphic scheme replaces the standard scheme in DE and is capable of mapping all 5 standard schemes (and more) used in literature. Adaptivity is realized on symbolic level changing the mathematical expression of the scheme based on success rates.

This paper is organized as follows. Section II introduces the Differential Evolution algorithm. A concept of polymorphic symbols and equations is introduced in section III. Based on this, in section IV the Polymorphic Differential Evolution approach is defined. In section V the conducted experiments with some results are presented. Finally, in section VI some conclusions are drawn.

II. Differential Evolution

Storn and Price introduced in [4], [5], [6] a real-valued vector based evolutionary algorithm for optimization in continuos spaces they called Differential Evolution (DE). The main idea employed is not to mutate vector components by simply replacing their values by random values. Instead, two population mates are randomly selected whose weighted difference is added to a third randomly selected population member creating this way a mutant vector. Then, either a two-point crossover ([4]) or a multi-point crossover ([6]) is performed

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

between the mutant vector and the current population member being considered. At the end, the new created offspring vector (trial vector) replaces the current considered vector in the next generation, iff its fitness is better. Otherwise, the trial vector is discarded.

Notation. Let D be the dimension of the problem (i.e., the dimension of the search space $\mathbb{R}^D$), NP the number of vectors and X_G the set of vectors $X_G = \{ x_{1,G}, \cdots, x_{NP,G} \}$ of generation G. Each vector $x_{i,G} \in X_G$ is an element of the search space ($x_{i,G} \in \mathbb{R}^D$).

Typically, an upper and lower bound is defined for each component j: $(x)_j^{\mathbf{L}}$ (lower bound) and $(x)_j^{\mathbf{U}}$ (upper bound). Thus, for all indices $j \in \{1, \cdots, D\}$ we have that $(x_{i,G})_j \in [(x)_j^{\mathbf{L}}, (x)_j^{\mathbf{U}}]$.

The best vector of generation G (global best) is denoted by $x_{best,G}$. To compute the objective value (or fitness) of a vector, a fitness function $F : \mathbb{R}^D \to \mathbb{R}$ is used.

Initialization. Firstly, the generation counter G is set to 0. Then, the initial population X_0 is created by randomly creating NP vectors $x_{i,0}$. For this, each component $(x_{i,0})_j$ of each vector is taken from $U((x)_j^{\mathbf{L}}, (x)_j^{\mathbf{U}})$.

Iteration. The iteration works as follows. Firstly, for each vector of the population a certain number of population mates are selected randomly (typically 3 or 5). The number of these selected mates depends on the scheme used to compute a mutant vector v_i. A well-known standard scheme using 3 population mates is shown in Eq. (1).

$$v_i = x_{r_1} + F(x_{r_2} - x_{r_3}) \tag{1}$$

All selected mates need to be mutually different and also to be different from the current vector x_i. Secondly, the mutant vector v_i is computed using a scheme. Thirdly, a crossover is performed between the mutant and the current vector leading to a trial vector u_i. Finally, this trial vector replaces the current vector, iff its fitness is better. In pseudo-code this looks as follows:

1. **for all** vectors $x_{i,G} \in X_G$ **do**:
   - 1.1. Determine indices $r_1, r_2, r_3, \cdots \sim U(1, NP)$ such that $i \neq r_1 \neq r_2 \neq r_3 \neq \cdots$
   - 1.2. Compute mutant vector v_i with a scheme, e.g., Eq. (1)
   - 1.3. Create trial vector u_i via crossover between current and mutant vector.
     - 1.3.1. Determine a random number for each vector component: $r \sim U(0,1)^D$
     - 1.3.2. Randomly select an index to ensure that at least one component undergoes crossover: $d \sim U(1, D)$
     - 1.3.3. $u_i = ((u_i)_1, \cdots, (u_i)_j, \cdots, (u_i)_D)^T$

$$(u_i)_j = \begin{cases} (v_i)_j & , \quad r_j \leq CR \vee j = d \\ (x_i)_j & , \quad r_j > CR \wedge j \neq d \end{cases}$$

   - 1.4. Decide, which one goes to next generation:

$$x_{i,G+1} = \begin{cases} u_i & , \quad F(u_i) < F(x_{i,G}) \\ x_{i,G} & , \quad \text{otherwise} \end{cases}$$

2. $G \leftarrow G + 1$

There are three user-defined parameters, which are application-dependent: (a) the mutation factor F as a real-valued constant from $[0,2]$ (but Storn also suggests in [6] that $F > 0$), (b) the crossover probability $CR \in [0,1]$, which is applied to each vector component separately and (c) the population size NP. In [6] the authors suggest to use the following parameters as a first try: $F = 0.5$, $CR = 0.1$ and $5 \cdot D \leq NP \leq 10 \cdot D$.

Variants. There are several well-known schemes for computing the mutant vectors. In this work the following schemes were used as introduced in [5] (names are overtaken):

DE/rand/1 :	$v_i = x_{r_1} + F(x_{r_2} - x_{r_3})$
DE/best/1 :	$v_i = x_{best} + F(x_{r_2} - x_{r_3})$
DE/rand-to-best :	$v_i = x_{r_1} + \lambda(x_{best} - x_{r_1}) + F(x_{r_2} - x_{r_3})$
DE/current-to-rand :	$v_i = x_i + \lambda(x_{r_1} - x_i) + F(x_{r_2} - x_{r_3})$
DE/current-to-best :	$v_i = x_i + \lambda(x_{best} - x_i) + F(x_{r_1} - x_{r_2})$

Parameter F has the same meaning as introduced in the iteration part. The λ parameter controls the greediness of the appropriate scheme. Storn recommends in [5] to set $\lambda = F$.

III. Polymorphic Symbols

In this work a symbol means each object or variable used in terms or equations as for instance x, y and z in $x + y = z$. A *Polymorphic Symbol* is a more abstract symbol, a 'meta-symbol', representing a finite set of symbols. To avoid term confusions, symbols being used in terms or equations are called variables, hereafter. Let $V =< v_1, \cdots, v_N >$ denote an ordered list of N variables v_i. Furthermore, let $P =< p_1, \cdots, p_N >$ denote an ordered list of properties, whereby property p_i corresponds to variable v_i.

Definition 3.1 (Polymorphic Symbol): A polymorphic symbol $\mathcal{S} = (V_\mathcal{S}, P_\mathcal{S}, C_\mathcal{S})$ is comprised of a list of variables $V_\mathcal{S}$, their corresponding properties $P_\mathcal{S}$ as well as a determination procedure $C_\mathcal{S} : \{\mathcal{S}_i\} \mapsto \{v_1, \cdots, v_N\}$. Procedure $C_\mathcal{S}$ gets a polymorphic symbol and returns a valid variable out of $V_\mathcal{S}$. A variable $C_\mathcal{S}(\mathcal{S})$ is called a configuration of $\mathcal{S}$.

Example. Lets define a polymorphic symbol $\mathcal{P} = (V_\mathcal{P} =< x, y, z >, P_\mathcal{P} =< 1,0,0 >, C_\mathcal{P})$. Furthermore, lets determine a variable by $C_\mathcal{P}(\mathcal{P}) = \arg\max_{v_i \in \{v_1, \cdots, v_N\}}(p_i)$. This procedure returns the variable whose corresponding property has the highest value. The configuration of $\mathcal{P}$ can now be controlled by manipulating its properties $P_\mathcal{P}$. If we set $P_\mathcal{P} =< 1,0,0 >$, then $C_\mathcal{P}(\mathcal{P}) = v_1 = x$. For $P_\mathcal{P} =< 0,1,0 >$ we obtain $C_\mathcal{P}(\mathcal{P}) = v_2 = y$. Finally, $P_\mathcal{P} =< 0,0,1 >$ leads to $C_\mathcal{P}(\mathcal{P}) = v_3 = z$.

Polymorphic Equations. It is possible to define *Polymorphic Equations*, if at least one variable is replaced by a polymorphic symbol. Lets take the example equation $x + y = z$ and lets replace the first variable by the configuration of polymorphic symbol $\mathcal{P}$:

$$C_\mathcal{P}(\mathcal{P}) + y = z \tag{2}$$

Depending on $P_\mathcal{P}$ three different equations are now possible: $x + y = z$, $y + y = z$ and $z + y = z$. If all variables are replaced by polymorphic symbols $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ (defined similarily to

$\mathcal{P}$) we get

$$C_{\mathcal{P}}(\mathcal{P}_1) + C_{\mathcal{P}}(\mathcal{P}_2) = C_{\mathcal{P}}(\mathcal{P}_3)$$

with $3^3 = 27$ different equation configurations. Note that the number of equation configurations is not necessarily the number of different mathematical expressions. Lets assume two different configurations (a) $C_{\mathcal{P}}(\mathcal{P}_1) = x$, $C_{\mathcal{P}}(\mathcal{P}_2) = y$, $C_{\mathcal{P}}(\mathcal{P}_3) = z$ and (b) $C_{\mathcal{P}}(\mathcal{P}_1) = y$, $C_{\mathcal{P}}(\mathcal{P}_2) = x$, $C_{\mathcal{P}}(\mathcal{P}_3) = z$ leading to the equations $x + y = z$ and $y + x = z$, respectively. Iff the $+$ operator used is commutative, then both equation configurations lead to the same mathematical expression.

The introduced concept of polymorphic symbols / equations allows for adaptive equations. A method is able to adapt its equations by itself by changing the properties of the used polymorphic symbols.

IV. Polymorphic Differential Evolution

The Polymorphic Differential Evolution (PolyDE) concept replaces the typically used schemes by a polymorphic scheme. A polymorpic scheme is a scheme, which uses only polymorphic variables (besides some scalars). Let $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_4$ and $\mathcal{P}_5$ be polymorphic symbols defined as:

$$\mathcal{P}_i = (V =< x_i, x_{best}, x_{r_i} >, P_{\mathcal{P}_i} =< 1,1,1 >, C) \qquad (3)$$

Each $\mathcal{P}_i$ uses the same variables and determination procedure. But each $\mathcal{P}_i$ has its own distinct property $P_{\mathcal{P}_i}$, since all variables are meant to be independently modifiable.

In PolyDE the properties $P_{\mathcal{P}_i}$ are interpreted as accumulative histograms. To get the configurations, these histograms are sampled using roulette-wheel sampling. That is, $C(\mathcal{P}_i)$ returns variable v_j with probability $\frac{(P_{\mathcal{P}_i})_j}{\sum_k (P_{\mathcal{P}_i})_k}$. Thus, the higher a histogram bin $(P_{\mathcal{P}_i})_j$, the higher the probability that the corresponding variable v_j is the configuration determined for a given polymorphic symbol.

The polymorphic scheme to be used in step 1.2. of the pseudo-code in section II is defined as:

$$v_i = C(\mathcal{P}_1) + \lambda(C(\mathcal{P}_2) - C(\mathcal{P}_3)) + F(C(\mathcal{P}_4) - C(\mathcal{P}_5)) \qquad (4)$$

Note that all used standard schemes as given in section II are special cases of this polymorphic scheme. Even the shorter schemes without the λ term can be mapped, if $C(\mathcal{P}_2) = C(\mathcal{P}_3)$. All together this polymorphic scheme is able to realize $3^5 = 243$ different schemes. (Well, the question, whether configurations with $C(\mathcal{P}_2) = x_i = C(\mathcal{P}_3)$ are considered equivalent to $C(\mathcal{P}_2) = x_{best} = C(\mathcal{P}_3)$ and thus need to be subtracted, is left to philosophers.) For instance, if one wants to realize the standard scheme as given in Eq. (1), then one could set the histograms the following way: $P_{\mathcal{P}_1} =< 0,0,1 >$, $P_{\mathcal{P}_2} =< 1,0,0 >$, $P_{\mathcal{P}_3} =< 1,0,0 >$, $P_{\mathcal{P}_4} =< 0,0,1 >$ and $P_{\mathcal{P}_5} =< 0,0,1 >$.

Initialization. PolyDE is initialized the same way as standard DE (see section II). Additionally, all bins of the accumulative histograms of all polymorphic symbols are set to 1 ($P_{\mathcal{P}_1} = P_{\mathcal{P}_2} = P_{\mathcal{P}_3} = P_{\mathcal{P}_4} = P_{\mathcal{P}_5} =< 1,1,1 >$). This way all symbols have the same probability to be chosen.

Iteration. Basically, the iteration is the same as in standard DE. What is different is the integration of the accumulative histograms as well as the polymorphic scheme. Algorithm 1 outlines the whole iteration step: The first step is to initialize the histograms for accumulating successful configurations denoted as $S_{\mathcal{P}_i}$. All successful trial vectors of an iteration are considered. Then, for all individuals of the current generation a mutant vector is computed. For this, five population mates are selected randomly $(r_1, \cdots, r_5)$ being mutually different. These five randomly chosen individuals $(x_{r_1,G}, \cdots, x_{r_5,G})$ together with the current individual $x_{i,G}$ and the global best $x_{best,G}$ are needed for the polymorphic symbols (remember the definiton of V in Eq. (3)). Afterwards, the actual configuration of all 5 polymorphic symbols are determined using $C(\cdot)$. These are hold in variables for the case of success to be able to count the right symbols (a second call to $C(\cdot)$ could deliver another symbol caused by the sampling process). Then, the polymorphic scheme is now used to compute the mutant vector v_i. After this, the trial vector u_i can be computed. Because for each component of the vector an own decision is to make whether or not to crossover, a random vector r is determined. Additionally, a random index d is determined to ensure that at least one component will undergo crossover. With r and d the crossover between the mutant and current vector is computed. In case that the obtained trial vector has a better fitness it is overtaken to the next generation, otherwise it is discarded. This implements a greedy strategy. If the trial vector was successful, i.e., its fitness is better, then the configuration $c_1, \cdots, c_5$ used for the polymorphic scheme is counted. For this, the appropriate histogram bins of the success histograms $S_{\mathcal{P}_1}, \cdots, S_{\mathcal{P}_5}$ are incremented. To determine the index number of the bin for a given symbol in configuration c_i, the $indexof(\cdot)$ operator is used. This operator returns 1 if $c_i = x_i$, 2 if $c_i = x_{best}$ and so on. Finally, after all individuals were processed, the success histograms are added to the accumulative histograms $P_{\mathcal{P}_1}, \cdots, P_{\mathcal{P}_5}$. This accumulates all successful configurations over all iterations realizing this way the adaptivity. The last step is to switch the generation counter G to the next one.

V. Experiments

To evaluate the capabilities of PolyDE, four typical benchmark functions as known from literature (Sphere, Rosenbrock, Rastrigin, Griewank), representing all four combinations of unimodal/multimodal with/without dependencies between the variables, were chosen.

Sphere is the following simple unimodal function without dependencies between the variables:

$$\begin{aligned} f(< x_i >) &= \sum_{i=1}^{n} x_i^2 \\ (X_{min}, X_{max}) &:= (-5.12, 5.12) \\ \text{Global minimum} &: f(< 0, \cdots, 0 >) = 0 \end{aligned}$$

Rosenbrock is also a unimodal function but it is a bit more difficult as Sphere, because there are some dependencies

Algorithm 1 Iteration of PolyDE

// Initialize histograms to count successful polymorphic symbols
$S_{\mathcal{P}_1} \leftarrow < 0,0,0 >$
$S_{\mathcal{P}_2} \leftarrow < 0,0,0 >$
$S_{\mathcal{P}_3} \leftarrow < 0,0,0 >$
$S_{\mathcal{P}_4} \leftarrow < 0,0,0 >$
$S_{\mathcal{P}_5} \leftarrow < 0,0,0 >$

// Update each individual in population
for all $x_{i,G} \in X_G$ **do**

// Compute mutant vector v_i
$r_1, r_2, r_3, r_4, r_5 \sim U(1, NP)$ such that $i \neq r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5$

$c_1 = C(\mathcal{P}_1)$ *// Determine configurations of polymorphic symbols*
$c_2 = C(\mathcal{P}_2)$
$c_3 = C(\mathcal{P}_3)$
$c_4 = C(\mathcal{P}_4)$
$c_5 = C(\mathcal{P}_5)$

$v_i = c_1 + \lambda(c_2 - c_3) + F(c_4 - c_5)$

// Create trial vector u_i *via crossover*
$r \sim U(0,1)^D$ *// Determine random number for each vector component*

$d \sim U(1, D)$ *// Random index to ensure that at least one component undergoes crossover*

$$u_i = (\ (u_i)_1, \cdots, (u_i)_j = \begin{cases} (v_i)_j & , \quad r_j \leq CR \vee j = d \\ (x_i)_j & , \quad r_j > CR \wedge j \neq d \end{cases} , \cdots, (u_i)_D\ )^T$$

// Overtake if better (Greedy Strategy)
if $F(u_i) < F(x_{i,G})$ **then**
$x_{i,G+1} = u_i$

$(S_{\mathcal{P}_1})_{indexof(c_1)} \leftarrow (S_{\mathcal{P}_1})_{indexof(c_1)} + 1$ *// Update success histograms*
$(S_{\mathcal{P}_2})_{indexof(c_2)} \leftarrow (S_{\mathcal{P}_2})_{indexof(c_2)} + 1$
$(S_{\mathcal{P}_3})_{indexof(c_3)} \leftarrow (S_{\mathcal{P}_3})_{indexof(c_3)} + 1$
$(S_{\mathcal{P}_4})_{indexof(c_4)} \leftarrow (S_{\mathcal{P}_4})_{indexof(c_4)} + 1$
$(S_{\mathcal{P}_5})_{indexof(c_5)} \leftarrow (S_{\mathcal{P}_5})_{indexof(c_5)} + 1$
else
$x_{i,G+1} = x_{i,G}$
end if

end for

// Accumulate success histograms
$P_{\mathcal{P}_1} \leftarrow P_{\mathcal{P}_1} + S_{\mathcal{P}_1}$
$P_{\mathcal{P}_2} \leftarrow P_{\mathcal{P}_2} + S_{\mathcal{P}_2}$
$P_{\mathcal{P}_3} \leftarrow P_{\mathcal{P}_3} + S_{\mathcal{P}_3}$
$P_{\mathcal{P}_4} \leftarrow P_{\mathcal{P}_4} + S_{\mathcal{P}_4}$
$P_{\mathcal{P}_5} \leftarrow P_{\mathcal{P}_5} + S_{\mathcal{P}_5}$

$G \leftarrow G + 1$

between the variables. It's defined as this:

$$\begin{aligned} f(< x_i >) &= \sum_{i=1}^{n-1} (100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \\ (X_{min}, X_{max}) &:= (-2.048, 2.048) \\ \text{Global minimum} &: f(< 1, \cdots, 1 >) = 0 \end{aligned}$$

Rastrigin is the following multimodal function without dependencies between the variables:

$$\begin{aligned} f(< x_i >) &= 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)) \\ (X_{min}, X_{max}) &:= (-5.12, 5.12) \\ \text{Global minimum} &: f(< 0, \cdots, 0 >) = 0 \end{aligned}$$

Griewank is a multimodal function with strong dependencies between the variables and is defined as follows:

$$\begin{aligned} f(< x_i >) &= 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) \\ (X_{min}, X_{max}) &:= (-600, 600) \\ \text{Global minimum} &: f(< 0, \cdots, 0 >) = 0 \end{aligned}$$

For each of these benchmark functions, the PolyDE approach as well as the standard DE method with all 5 schemes were conducted. We need to compare to all schemes, since our main aim is to get rid of the scheme selection process. This makes only sense, if PolyDE has at least a comparable performance compared to all schemes considered. Furthermore, the SADE scheme selection concept as introduced in [3] was also considered. Because all compared methods use fix strategy parameters, the SADE parameters were also set to the same fixed values to avoid side effects by the adaptive parameters.

The experiments were conducted with a dimension of 30 for each benchmark function. For each benchmark function 100 independent runs with 1000 iterations were performed. All methods (PolyDE, standard DE, SADE) used the standard parameters: $F = 0.5$, $\lambda = 0.5$ and $CR = 0.1$. The size of population was set according to the $5 \cdot D$ rule ($NP = 150$). In Table I the averaged results are presented.

The Sphere benchmark function is solved by all methods in a similar manner. All have a perfect hit rate of 100%. The only differences are the number of needed evaluations needed to reach the goal. At this point the PolyDE method is slightly better than all the others.

For Rosenbrock all methods produced a comparable fitness as well as number of evaluation steps. The performance of PolyDE is ranked in the middle.

The results for Rastrigin are quite interesting. On the one hand the `DE/rand-to-best` method produces reliably a very good fitness value, but on the other hand it never reached the perfect solution. Contrary, PolyDE is the only method being able to produce perfect solutions in 79%, but in non-perfect cases it ranks only in the middle. Thus, both methods are considered to be winners, although PolyDE needs fewer evaluation steps to reach its goal.

The `DE/rand-to-best` method is the clear winner for the Griewank function. The second and third places go to PolyDE and the `DE/current-to-best` method. Well, which one of both is the second best depends on your criteria. If fitness is of greater interest, then it is `DE/current-to-best`. But if the number of evaluation steps is more important, then it is better to choose PolyDE. In both cases PolyDE lies in the upper middle region.

Considered over all benchmark functions it can be stated that PolyDE is never one of the worse methods. In two cases (Sphere, Rastrigin) it is among the winners and in the other two cases it is in the (upper) middle region. Note that PolyDE belongs only to the winners for both functions having variables without dependencies. For the functions with dependencies it lies only in the middle. Furthermore, it seems that PolyDE is advantageous for multimodal functions (especially those without dependencies in the variables). Additionally, the results in Table I show the differences in performance that can be caused by pre-defined schemes. The same scheme can have worse performance in one case and a better performance in another case. But it is not known in advance, whether having the one or other case. In particular this is true for unknown functions. In PolyDE performance appears to be more fair and we always have at least average performance.

VI. Conclusions

In this paper a new concept called Polymorphic Differential Evolution (PolyDE) was introduced and applied to four typical benchmark functions. Based on a concept of polymorphic equations, the standard DE scheme was redefined to become a polymorphic scheme. The mathematical expression of this polymorphic scheme is changed on symbolic level based on accumulative histograms and roulette-wheel sampling introducing this way adaptivity. This polymorphic scheme is able to map all 5 standard schemes known from literature as special cases. In this sense it is a generalization of DE schemes.

The main aim of this work was to get rid of the scheme selection process in DE. Since the performance of PolyDE is ranked either in the top or the middle region it can be used as alternative to the standard DE schemes fulfilling this way the main aim. Additionally, in 3 of the 4 benchmark functions PolyDE needs fewer evaluation steps to reach its goal. The best performance was obtained for both multimodal functions.

References

[1] J. Brest, V. Zumer, M.S. Maucec, *Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization*, Proceedings of the 2006 IEEE Congress on Evolutionary Computation, pp. 215-222, 2006

[2] R. Gämperle, S.D. Müller, P. Koumoutsakos, *A Parameter Study for Differential Evolution*, Proc. Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, Interlaken, Switzerland, 2002

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

[4] R. Storn and K. Price, *Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces*, Univ. California, Berkeley, ICSI, Technical Report TR-95-012, March 1995 Downloadable from ftp://ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.pdf

Benchmark	DE method	Avg. Fitness	s.d.	Perfect hits (%)	Avg. Gen.	Avg. Num. Eval.
Sphere	PolyDE	0.0	0.0	100	491.98	73650
Sphere	DE/rand-to-best	0.0	0.0	100	525.01	78750
Sphere	DE/best/1	0.0	0.0	100	566.47	84900
Sphere	DE/current-to-best	0.0	0.0	100	630.85	94500
Sphere	SADE	0.0	0.0	100	713.91	106950
Sphere	DE/current-to-rand	0.0	0.0	100	719.06	107850
Sphere	DE/rand/1	0.0	0.0	100	938.85	140700
Rosenbrock	DE/best/1	24.5458302728	1.5933457755	0	995.28	149250
Rosenbrock	SADE	24.7699151755	0.7764666635	0	994.85	149100
Rosenbrock	DE/rand/1	24.7730584981	0.7873764301	0	990.37	148500
Rosenbrock	DE/current-to-best	25.1630688781	0.7651498309	0	995.22	149250
Rosenbrock	PolyDE	25.4622133323	0.6859917720	0	996.04	149400
Rosenbrock	DE/rand-to-best	25.5967563902	0.5071396419	0	995.75	149250
Rosenbrock	DE/current-to-rand	25.6343048616	0.4984283947	0	995.28	149250
Rastrigin	PolyDE	0.1727297746	0.4469271528	79	746.17	111900
Rastrigin	DE/rand-to-best	0.0000000001	0.0000000001	0	998.33	149700
Rastrigin	DE/best/1	0.0497479541	0.2168462989	0	998.33	149700
Rastrigin	SADE	0.0771708347	0.0539210534	0	997.61	149550
Rastrigin	DE/current-to-best	0.2930089132	0.1387341269	0	996.29	149400
Rastrigin	DE/current-to-rand	1.9523013721	0.5486120874	0	993.70	148950
Rastrigin	DE/rand/1	10.1586804154	1.7018862812	0	987.79	148050
Griewank	DE/rand-to-best	0.0	0.0	100	667.59	100050
Griewank	PolyDE	0.0000985728	0.0009807874	99	631.06	94650
Griewank	DE/current-to-best	0.000000000001	0.000000000002	98	906.73	135900
Griewank	SADE	0.000000000001	0.000000000001	98	936.21	140400
Griewank	DE/best/1	0.0003450665	0.001708296977	96	734.34	110100
Griewank	DE/current-to-rand	0.000000000012	0.000000000014	0	997.33	149550
Griewank	DE/rand/1	0.0000000009	0.0000000010	0	997.18	149550

TABLE I

RESULTS FOR ALL BENCHMARK PROBLEMS AVERAGED OVER 100 INDEPENDENT RUNS. THE COLUMN 'AVG. FITNESS' IS THE BEST FITNESS VALUE REACHED ON AVERAGE, 'S.D.' THE APPROPRIATE STANDARD DEVIATION, 'PERFECT HITS (%)' THE PERCENTAGE OF PERFECT SOLUTIONS ($F(x_{i,G}) = 0$), 'AVG. GEN.' THE AVERAGE NUMBER OF NEEDED GENERATIONS FOR THE BEST SOLUTION OF A RUN AND 'AVG. NUM. EVAL.' THE NUMBER OF NEEDED EVALUATIONS ON AVERAGE. WITHIN EACH BENCHMARK FUNCTION THE METHODS ARE RANKED ACCORDING TO THE FOLLOWING CRITERIA (IN THIS ORDER): 'PERFECT HITS (%)', 'AVG. FITNESS', 'AVG. GEN.'.

[5] R. Storn, *On the Usage of Differential Evolution for Function Optimization*, In: 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996), Berkeley, pp. 519-523. IEEE, USA, 1996

[6] R. Storn and K. Price, *Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces*, Journal of Global Optimization, Volume 11, Number 4 / Dezember 1997, pp. 341 - 359, Springer Netherlands, 1997

[7] Y. Zhenyu, T. Ke, Y. Xin, *Self-adaptive differential evolution with neighborhood search*, Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 1110-1116, 2008