Tracking Differential Evolution Algorithms: An adaptive approach through multinomial distribution tracking with exponential forgetting

M. G. Epitropakis[1]*, D. K. Tasoulis[2], N. G. Pavlidis[3], V. P. Plagianakos[4], and M. N. Vrahatis[1]

[1] Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR-26110 Patras, Greece,
[2] Winton Capital Management, 1–5 St Mary Abbots Place, London SW8 6LS, U.K.,
[3] Department of Management Science, Lancaster University, LA1 4YX, U.K.,
[4] Department of Computer Science and Biomedical Informatics, University of Central Greece, GR-35100 Lamia, Greece,
mikeagn@math.upatras.gr, d.tasoulis@wintoncapital.com, n.pavlidis@lancaster.ac.uk, vpp@ucg.gr, vrahatis@math.upatras.gr

Abstract. Several Differential Evolution variants with modified search dynamics have been recently proposed, to improve the performance of the method. This work borrows ideas from adaptive filter theory to develop an "online" algorithmic adaptation framework. The proposed framework is based on tracking the parameters of a multinomial distribution to reflect changes in the evolutionary process. As such, we design a multinomial distribution tracker to capture the successful evolution movements of three Differential Evolution algorithms, in an attempt to aggregate their characteristics and their search dynamics. Experimental results on ten benchmark functions and comparisons with five state-of-the-art algorithms indicate that the proposed framework is competitive and very promising.

Keywords: Differential Evolution, Adaptation, Multinomial Distribution, Exponential forgetting

1 Introduction

The Differential Evolution (DE) algorithm is a population–based stochastic direct search method that utilizes concepts borrowed from the broad class of Evolutionary Algorithms. Several variants of the original DE algorithm have been recently proposed [1–4, 9–12, 14]. Nevertheless, a relatively small number of DE variants have exhibited substantial performance gains in a large number of real-world applications, and hence few variants have attracted the attention of the Evolutionary Computing research community. These variants successfully exploit different aspects of the DE algorithm, either by utilizing novel mutation strategies that exploit DE's exploratory/exploitative search power, or by incorporating adaptive schemes to capture the structure of the benchmark function

* Corresponding author

at hand. Representative examples of the former type include specialized mutation strategies [9], index neighborhood-based mutations [2], or proximity mutations [4]. Variants of the latter type include schemes such as parameter and strategy adaptation schemes [1, 5, 10, 14].

In this study, we borrow ideas from adaptive filter theory to develop an "online" algorithm adaptation technique and incorporate it in the Differential Evolution algorithm. The proposed framework uses three DE variants, namely JADE [14], jDE [1], and DEGL [2] algorithms. It allows each individual to randomly select amongst them to evolve at each time step. The probability of selecting each variant depends on its history of improving the population at each time step and thus guiding it to promising search regions. Extensive experimental results on 10 benchmark functions demonstrate that the proposed framework is very promising.

The rest of the paper is organized as follows: Section 2 briefly describes the multinomial distribution tracker along with its main characteristics. Its incorporation into the DE algorithm as a new algorithm adaptation framework is briefly described in Section 3. The paper ends with an experimental analysis of the proposed framework, a discussion and some pointers for future work.

2 Multinomial Distribution Tracking through exponential forgetting

This section briefly presents the multinomial distribution and its extension to include exponential forgetting. In the multinomial distribution each trial results in one out of a fixed and finite number K of possible outcomes, with probabilities $\theta_1, \theta_2, \ldots, \theta_K$ and N independent trials. The number of times outcome i was observed over the N trials, is represented by a random variable X_i. Thereby, the vector $X = (X_1, X_2, \ldots, X_K)$ follows a multinomial distribution with parameters N, θ, where $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ and probabilities: $P(X_1 = x_1, \ldots, X_K = x_K|\theta, N) = (N!)/(\prod_{i=1}^{K} x_i!) \prod_{i=1}^{K} \theta_i^{x_i}$. Based on a data sample D we can estimate the parameter $\widehat{\theta} = \theta(D)$ of a multinomial distribution through the Maximum Likelihood Estimation (MLE) procedure. Given a data sample D, the likelihood function can be defined as: $L(\theta; D) = p(D|\theta) = p(x_1, x_2, \ldots, x_K|\theta)$. The MLE estimator of the θ parameter can be easily calculated by applying Lagrange multipliers in the log-likelihood function. Therefore the MLE of the multinomial distribution can be obtained by the following form: $\widehat{\theta}_i^{\mathrm{ML}} = m_k/N$, where $m_k = \sum_{i=1}^{K} x_i$.

We make the assumption that the impact of each observation should be related to the time of observation. This is reasonable since the optimization procedure frequently changes phases through evolution. More recent information about the evolution phase is expected to be more relevant to the optimization procedure while earlier information should be gradually disregarded. In the current study, we develop a tracking framework that is based on the Recursive Least Squares (RLS) adaptive filter [7, 8]. To this end, we incorporate weights to the likelihood function and adopt the framework proposed in [8]. Given that a data sample appears as a signal or a data stream in time, $D = \{D_1, D_2, \ldots, D_t, \ldots\}$, where t denotes the current time step. We incorporate an exponential weighting factor in the log-likelihood function and produce a new likelihood which incorporates time, $L^{\lambda}(\theta|D_1, D_2, \ldots, D_t)$. As in the RLS filter, the new likelihood can be defined as: $L^{\lambda}(\theta|D_1, D_2, \ldots, D_t) = \sum_{j=1}^{t} \lambda^{t-j} L(\theta|D_1, \ldots, D_j) = L(\theta|D_t) + \lambda L(\theta|D_1, \ldots, D_{t-1})$, where $\lambda \in [0, 1]$ is a weighting factor which is also called the *forgetting factor*. The forgetting factor decreases the impact of past observations on the log-likelihood and thus the estimated parameters are able to adapt to changes. All data examples are assigned equal weights as λ increases to unity, while as λ decreases more recent data samples become more important.

Through the application of Lagrange multipliers we can obtain the MLE $\widehat{\theta}_i^{\mathrm{ML}_\lambda}$ according to the following equation:

$$\widehat{\theta}_i^{\mathrm{ML}_\lambda}(t) = \frac{n_i(t)}{\sum_{k=1}^{K} n_k(t)}. \tag{1}$$

where $n_i(t)$ represents the effective window width which can be recursively calculated through the following equation:

$$n_i(t) = \lambda n_i(t-1) + D_t^i, \tag{2}$$

for $t = 1, 2, \ldots$ and $n_i(0) = 0$, where D_t^i denotes the number of successes of outcome i at time t. If $\lambda = 1$ the aforementioned framework corresponds to the simple case of the $\widehat{\theta}_i^{\mathrm{ML}}$ MLE. Through this framework we can track the parameters of a multinomial distribution with the potential of forgetting the history of past observations in an exponential manner.

3 The Multinomial distribution-based Differential Evolution framework

In this section, we discuss the main concepts behind the proposed framework, namely the Multinomial distribution-based Differential Evolution (MultiDE). The proposed framework is based on the Differential Evolution algorithm (DE) [12]. DE is a population–based stochastic optimization method, which utilizes concepts borrowed from the broad class of Evolutionary Algorithms. For an optimization problem at hand defined in the real D–dimensional space $\mathbb{R}^D$, DE starts by initializing randomly a population of *NP*, potential solutions (*individuals*) in the optimization domain following a uniform probability distribution. The population is subsequently updated at each iteration, called *generation*, by means of three main evolutionary search operations, namely the *mutation*, *recombination*, and *selection* operators. The search operators efficiently shuffle information among the individuals, enabling the search for an optimum to focus on the most promising regions of the solution space. A thorough description of the DE algorithm can be found in [3, 4, 9, 12]

The proposed framework introduces two main concepts different from the standard DE. Initially, it probabilistically assigns to each individual one DE variant, chosen from a pool of K candidate algorithms. Subsequently, based on the individuals' movements, it adapts this probability over the evolutionary

Algorithm 1 The MultiDE algorithmic scheme

```
1: Initialize individuals of the population
2: Initialize the multinomial distribution tracker, for each algorithm $i : n_i(t_0) = 0$, $\widehat{\theta}_i^{\mathrm{ML}_\lambda}(t_0) = \frac{1}{K}$.
3: for each time step $t$ do
4:     for each individual $j$ in the population do
5:         Sample $k_{str}$ from the multinomial distribution with parameters $\widehat{\theta}_i^{\mathrm{ML}_\lambda}(t)$.
6:         Apply the mutation operator using algorithm $k_{str}$, $k_{str} \in$ {1) JADE, 2) jDE,
           3) DEGL}.
7:         Apply the binomial crossover operator
8:         Apply the selection operator
9:     end for
10:    Update the score of the $k_{str}$ strategy through Eq. (3)
11:    Update the multinomial distribution tracker through Eqs. (1)–(2)
12: end for
```

stages through the aforementioned multinomial distribution tracker. The remaining steps of the DE algorithm remain the same. In the current study we utilize a pool of $K = 3$ state-of-the-art DE variants that have efficiently tackled several real or artificial problem landscapes, namely the JADE [14], the jDE [1], and the DEGL [2] algorithms. Obviously any DE variant could be incorporated into the pool to enhance the exploratory and exploitative power of the proposed framework. Subsequently, one of the available algorithms is assigned to each individual based on a probability. This probability is adapted at each generation through the multinomial distribution tracker, based on the relative fitness improvement of each algorithm [5].

Specifically, let us assume that the ith algorithm $i \in \{1, 2, \ldots, K\}$, will evolve NP_i individuals, with $\sum_{i=1}^{K} NP_i = NP$. For each individual $j, j \in \{1, 2, \ldots, NP_i\}$ we assign a score based on its relative fitness improvement during the last generation according to $w_j = f_{\mathrm{b}est}|f^i_{\mathrm{p}arent} - f^i_{\mathrm{o}ffspring}|/f^i_{\mathrm{o}ffspring}$, where $f_{\mathrm{b}est}$ is the fitness of the best individual, $f^i_{\mathrm{p}arent}$ is the fitness of the ith individual before the evolution phase, while $f^i_{\mathrm{o}ffspring}$ is its fitness after the evolution phase [5]. The final score, Score(i), of each algorithm can be calculated according to the following formula:

$$\mathrm{Score}(i) = \mathrm{round}\left(100K \frac{\mathrm{W}(i)}{\sum_{i=1}^{K} \mathrm{W}(i)}\right), \tag{3}$$

where $\mathrm{W}(i) = w_{\min} + \sum_{j=1}^{NP_i} w_j$, $w_{\min} = 0.01$ is a small constant that prevents the extinction of an algorithm, in the case where the ith algorithm has not been selected in the previous generation. The rounding procedure as well as the multiplication by $100K$ will fix the score to the required integer value, by the multinomial distribution. The final score assists the algorithm which produces the higher relative fitness improvement in the last generation. Thereby, the multinomial distribution tracker learns from the current evolution stage and promotes the algorithm that is more likely to efficiently evolve the population to promising search regions. Having calculated the final scores, we estimate the probabilities of each algorithm by calculating the aforementioned maximum likelihood estimator $\widehat{\theta}_i^{\mathrm{ML}_\lambda}$, given by Eqs. (1) and (2). The main algorithmic scheme of the proposed framework is briefly demonstrated in Algorithm 1.

4 Experimental results

In this section we perform an experimental evaluation of the proposed approach. We employ ten high dimensional and scalable benchmark functions with different characteristics. The first six functions have been acquired from the recently CEC'2008 Special Session on Large Scale Global Optimization [13]. The remaining four test functions are hybrid composition functions, proposed recently in [6], and correspond to the $f_{16} - f_{19}$ functions of the suite. A detailed description of the benchmark functions can be found in [6, 13]. To demonstrate the efficiency of the proposed framework, we compare it with five state-of-the-art DE variants, namely the DEGL [2], the JADE [14], the jDE [1], the ODE [11], and the SADE [10] variant.

Throughout the experimental results section, all methods have been implemented with the default parameters settings as have been proposed in the literature. The population size has been kept fixed to $NP = 100$ individuals and for each simulation, a budget of $\max NFEs = 5000 \cdot D$ function evaluations has been employed [13]. Here we utilize the 50–dimensional versions of the aforementioned function set. To evaluate the performance of the considered algorithms we will use the *solution error measure*, or simply *error* [4]. Each algorithm was executed independently 50 times to obtain an estimation of the median (*Median*), the mean solution error (*Mean*), and its standard deviation (*St.D.*). Moreover to evaluate the statistical significance of the observed performance differences, we apply two-sided Wilcoxon rank sum tests between the proposed DE variants and the other DE variants. Here we have implemented three different forgetting factor values, $\lambda \in \{0.91, 0.99, 1\}$. The first two values force to forget the history of the strategy probabilities with either a fast or a slow rate, respectively, i.e. a sliding window size of $w \approx 11.1$, or $w \approx 100$ generations respectively. The sliding window can be approximated using the λ parameter, through: $w \approx 1/(1 - \lambda)$ [7].

Tables 1 and 2 report the experimental results on the 50–dimensional versions of the considered benchmark set. It can be clearly observed that the synergy of DE variants through the multinomial distribution tracker may result to an enhanced DE scheme, with a lot of potential. In general, for the majority of the considered functions, MultiDE exhibits either a significant performance enhancement, or an equally good performance in comparison to the other five DE variants. Only in three functions the proposed framework exhibits inferior performance in terms of median error values (f_2, f_3 and f_5). Substantial performance gains are mainly exhibited in the most challenging functions of the test suite, i.e. the hybrid composition functions ($f_6 - f_{10}$) and f_4. In these cases the proposed approaches significantly outperform all other DE variants. Comparing the non-forgetting, (MultiDE$_{\lambda=1.00}$), against the forgetting variants (MultiDE$_{\lambda=0.91}$ and MultiDE$_{\lambda=0.99}$), we can observe that in the majority of cases there is no significant performance difference. Only in f_8 and f_9 the forgetting variants exhibit significantly better behavior. However, in most cases the non-forgetting variants produce lower median and mean error values.

Generally, we have observed that the adaptation of the strategy probabilities behave differently based on the benchmark problem at hand as well as the

Table 1. Error values of the proposed DE framework, MultiDE, and five state-of-the-art DE variants on the first five 50–dimensional versions of the considered benchmark set (f_1–f_5).

Algorithm	Median	Mean	St.D.	NFE	Success	St. Sig.
			f_1 : Shifted Sphere Function			
DEGL	**0.000e+00**	**0.000e+00**	0.000e+00	3.230e+04	100.0	(=/=/=)
JADE	**0.000e+00**	**0.000e+00**	0.000e+00	4.363e+04	100.0	(=/=/=)
jDE	**0.000e+00**	**0.000e+00**	0.000e+00	1.535e+05	100.0	(=/=/=)
ODE	**0.000e+00**	**0.000e+00**	0.000e+00	1.194e+05	100.0	(=/=/=)
SADE	**0.000e+00**	**0.000e+00**	0.000e+00	7.779e+04	100.0	(=/=/=)
MultiDE$_{\lambda=0.91}$	**0.000e+00**	**0.000e+00**	0.000e+00	5.032e+04	100.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	**0.000e+00**	**0.000e+00**	0.000e+00	5.056e+04	100.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	**0.000e+00**	**0.000e+00**	0.000e+00	5.002e+04	100.0	(=/=/=)
			f_2 : Shifted Schwefel's Problem 2.21			
DEGL	1.430e+01	1.559e+01	5.660e+00	N/A	0.0	(+/+/+)
JADE	**1.115e+00**	**1.127e+00**	2.264e-01	N/A	0.0	(–/–/–)
jDE	2.220e+00	2.543e+00	1.581e+00	N/A	0.0	(–/–/–)
ODE	4.631e+00	6.115e+00	6.306e+00	N/A	0.0	(=/–/–)
SADE	2.925e+01	2.961e+01	3.300e+00	N/A	0.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	5.678e+00	6.471e+00	3.266e+00	N/A	0.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	6.161e+00	6.718e+00	3.269e+00	N/A	0.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	6.396e+00	6.765e+00	2.794e+00	N/A	0.0	(=/=/=)
			f_3 : Shifted Rosenbrock's Function			
DEGL	**0.000e+00**	**1.356e+00**	1.908e+00	1.834e+05	66.0	(–/–/–)
JADE	2.286e+00	9.801e+00	1.956e+01	2.284e+05	8.0	(=/=/=)
jDE	3.975e+01	4.698e+01	1.956e+01	N/A	0.0	(+/+/+)
ODE	4.683e+01	9.303e+04	4.997e+05	N/A	0.0	(+/+/+)
SADE	1.918e+01	2.553e+01	2.439e+01	N/A	0.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	3.015e+00	4.210e+00	9.579e+00	2.210e+05	2.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	3.278e+00	4.480e+00	9.818e+00	2.477e+05	2.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	2.873e+00	3.110e+00	3.023e+00	2.498e+05	2.0	(=/=/=)
			f_4 : Shifted Rastrigin's Function			
DEGL	1.492e+02	1.515e+02	2.531e+01	N/A	0.0	(+/+/+)
JADE	**0.000e+00**	**0.000e+00**	0.000e+00	2.217e+05	100.0	(=/=/=)
jDE	7.136e+01	7.198e+01	6.070e+00	N/A	0.0	(+/+/+)
ODE	3.651e+02	3.488e+02	4.078e+01	N/A	0.0	(+/+/+)
SADE	0.000e+00	5.680e+00	1.106e+01	2.305e+05	68.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.935e+05	100.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.962e+05	100.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	**0.000e+00**	**0.000e+00**	0.000e+00	2.158e+05	100.0	(=/=/=)
			f_5 : Shifted Griewank's Function			
DEGL	7.000e-03	1.382e-02	2.143e-02	3.251e+04	48.0	(+/+/+)
JADE	0.000e+00	1.020e-03	2.839e-03	4.470e+04	88.0	(=/=/=)
jDE	**0.000e+00**	**0.000e+00**	0.000e+00	1.538e+05	100.0	(–/–/–)
ODE	0.000e+00	1.200e-03	3.090e-03	1.212e+05	86.0	(=/=/=)
SADE	0.000e+00	5.760e-03	1.041e-02	7.681e+04	64.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	0.000e+00	6.800e-04	2.369e-03	5.018e+04	92.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	0.000e+00	1.480e-03	3.694e-03	5.118e+04	84.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	0.000e+00	1.320e-03	3.766e-03	5.041e+04	88.0	(=/=/=)

evolution phase. This indicates that the forgetting factor values should adapt through different evolution phases. This is a very interesting research area that we intend to extensively study in the future.

5 Conclusions

Recent Differential Evolution variations suggest that the advantages of several DE variants can be exploited by integrating them in adaptive schemes. We attempt to exploit the characteristics of different DE variants in an attempt to

Table 2. Error values of the proposed DE framework, MultiDE, and five state-of-the-art DE variants on the last five 50–dimensional versions of the considered benchmark set (f_6–f_{10}).

Algorithm	Median	Mean	St.D.	NFE	Success	St. Sig.
			f_6 : Shifted Ackley's Function			
DEGL	2.901e+00	3.001e+00	6.213e-01	N/A	0.0	(+/+/+)
JADE	**0.000e+00**	**0.000e+00**	0.000e+00	6.353e+04	100.0	(=/=/=)
jDE	**0.000e+00**	**0.000e+00**	0.000e+00	2.281e+05	100.0	(=/=/=)
ODE	0.000e+00	2.460e-03	1.739e-02	1.753e+05	98.0	(=/=/=)
SADE	0.000e+00	3.442e-01	5.674e-01	1.161e+05	72.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	**0.000e+00**	**0.000e+00**	0.000e+00	7.363e+04	100.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	**0.000e+00**	**0.000e+00**	0.000e+00	7.336e+04	100.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	0.000e+00	1.758e-02	1.243e-01	7.362e+04	98.0	(=/=/=)
			f_7 : Hybrid Composition Function 1 (f_{16} [6])			
DEGL	8.144e+01	7.839e+01	3.066e+01	N/A	0.0	(+/+/+)
JADE	1.167e-05	2.697e-05	5.025e-05	N/A	0.0	(+/+/+)
jDE	9.126e-05	9.085e-05	2.451e-05	N/A	0.0	(+/+/+)
ODE	5.736e-03	6.463e-03	3.008e-03	N/A	0.0	(+/+/+)
SADE	3.124e-02	2.524e-01	1.503e+00	2.188e+05	12.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	1.559e-13	1.426e-03	6.143e-03	1.749e+05	94.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	**1.165e-13**	2.146e-04	1.517e-03	1.741e+05	98.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	1.788e-13	**2.167e-13**	1.990e-13	1.767e+05	100.0	(=/=/=)
			f_8 : Hybrid Composition Function 2 (f_{17} [6])			
DEGL	1.157e+02	1.151e+02	3.575e+01	N/A	0.0	(+/+/+)
JADE	8.797e+00	9.285e+00	1.551e+00	N/A	0.0	(+/+/+)
jDE	6.427e+00	6.320e+00	9.796e-01	N/A	0.0	(+/+/+)
ODE	8.416e+00	8.591e+00	8.773e-01	N/A	0.0	(+/+/+)
SADE	4.013e-01	1.241e+00	3.015e+00	N/A	0.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	1.535e-01	**3.250e-01**	7.725e-01	N/A	0.0	(=/=/–)
MultiDE$_{\lambda=0.99}$	**1.213e-01**	3.729e-01	9.330e-01	N/A	0.0	(=/=/–)
MultiDE$_{\lambda=1.00}$	1.906e-01	3.587e-01	1.144e+00	N/A	0.0	(+/+/=)
			f_9 : Hybrid Composition Function 3 (f_{18} [6])			
DEGL	3.777e+01	3.621e+01	1.070e+01	N/A	0.0	(+/+/+)
JADE	2.792e-01	2.787e-01	3.346e-02	N/A	0.0	(+/+/+)
jDE	1.098e-01	1.082e-01	3.737e-02	N/A	0.0	(+/+/+)
ODE	5.347e+00	7.079e+00	5.950e+00	N/A	0.0	(+/+/+)
SADE	9.096e-02	9.885e-02	7.226e-02	N/A	0.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	**4.158e-04**	**6.529e-04**	8.106e-04	N/A	0.0	(=/–/–)
MultiDE$_{\lambda=0.99}$	1.001e-03	2.019e-03	2.570e-03	N/A	0.0	(+/=/–)
MultiDE$_{\lambda=1.00}$	4.474e-02	5.518e-02	2.886e-02	N/A	0.0	(+/+/=)
			f_{10} Hybrid Composition Function 4 (f_{19} [6])			
DEGL	1.175e+01	1.114e+01	2.671e+00	N/A	0.0	(+/+/+)
JADE	**0.000e+00**	1.823e-01	3.965e-01	5.144e+04	80.0	(=/=/=)
jDE	8.186e-15	**9.568e-15**	5.738e-15	1.662e+05	100.0	(+/+/+)
ODE	5.526e-17	3.022e-01	6.365e-01	1.535e+05	78.0	(+/+/+)
SADE	2.100e+00	2.132e+00	1.442e+00	8.487e+04	12.0	(+/+/+)
MultiDE$_{\lambda=0.91}$	**0.000e+00**	2.104e-01	4.452e-01	5.848e+04	76.0	(=/=/=)
MultiDE$_{\lambda=0.99}$	**0.000e+00**	2.816e-01	4.340e-01	5.850e+04	66.0	(=/=/=)
MultiDE$_{\lambda=1.00}$	**0.000e+00**	3.236e-01	6.636e-01	5.793e+04	72.0	(=/=/=)

improve their performance. Borrowing ideas from adaptive filter theory we develop an "online" algorithmic adaptation framework. The proposed framework is based on tracking the parameters of a multinomial distribution to capture the potentially changing probabilities of success of the different algorithms involved.

Experimental results on 10 benchmark functions demonstrate that the proposed framework is very promising. For the majority of the tested cases, it exhibits great performance gains against other five DE variants. The most appropriate degree of forgetting depends on the evolution stage, as well as the problem. It would be interesting to further study its impact and develop an adaptive forgetting factor scheme.

Acknowledgments. M.G. Epitropakis and M.N. Vrahatis would like to thank the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: "Heracleitus II: Investing in knowledge society through the European Social Fund" for financially supporting this work.

References

1. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-Adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (2006)
2. Das, S., Abraham, A., Chakraborty, U.K., Konar, A.: Differential evolution using a Neighborhood-Based mutation operator. IEEE Transactions on Evolutionary Computation 13(3), 526–553 (2009)
3. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1), 4–31 (2011)
4. Epitropakis, M.G., Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Enhancing differential evolution utilizing proximity-based mutation operators. IEEE Transactions on Evolutionary Computation 15(1), 99–119 (2011)
5. Gong, W., Álvaro Fialho, Cai, Z., Li, H.: Adaptive strategy selection in differential evolution for numerical optimization: An empirical study. Information Sciences 181(24), 5364–5386 (2011)
6. Lozano, M., Molina, D., Herrera, F.: Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. Soft Computing - A Fusion of Foundations, Methodologies and Applications 15(11), 2085–2087 (2011)
7. Niedzwiecki, M.: Identification of Time-varying Processes. John Wiley & Sons, New York (2000)
8. Pavlidis, N.G., Tasoulis, D.K., Adams, N.M., Hand, D.J.: λ-perceptron: An adaptive classifier for data streams. Pattern Recognition 44(1), 78–96 (2011)
9. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2005)
10. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13(2), 398–417 (2009)
11. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-Based differential evolution. IEEE Transactions on Evolutionary Computation 12(1), 64–79 (2008)
12. Storn, R., Price, K.: Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
13. Tang, K., *et al.*: Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Tech. rep., Nature Inspired Computation and Applications Laboratory, USTC, China (2007)
14. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13(5), 945–958 (2009)