Tracking Particle Swarm Optimizers: An adaptive approach through multinomial distribution tracking with exponential forgetting

M.G. Epitropakis*, D.K. Tasoulis†, N.G. Pavlidis‡, V.P. Plagianakos§, and M.N. Vrahatis*
*Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR-26110 Patras, Greece, Email: {mikeagn,vrahatis}@math.upatras.gr
†Winton Capital Management, 1–5 St Mary Abbots Place, London SW8 6LS, U.K., Email: d.tasoulis@wintoncapital.com
‡Department of Management Science, Lancaster University, Lancaster LA1 4YX, U.K., Email: n.pavlidis@lancaster.ac.uk
§Department of Computer Science and Biomedical Informatics, University of Central Greece, GR-35100 Lamia, Greece, Email: vpp@ucg.gr

***Abstract*—An active research direction in Particle Swarm Optimization (PSO) is the integration of PSO variants in adaptive, or self-adaptive schemes, in an attempt to aggregate their characteristics and their search dynamics. In this work we borrow ideas from adaptive filter theory to develop an "online" algorithm adaptation framework. The proposed framework is based on tracking the parameters of a multinomial distribution to capture changes in the evolutionary process. As such, we design a multinomial distribution tracker to capture the successful evolution movements of three PSO variants. Extensive 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. On the majority of tested cases, the proposed framework achieves substantial performance gain, while it seems to identify accurately the most appropriate algorithm for the problem at hand.**

I. Introduction

The Particle Swarm Optimization (PSO) algorithm belongs to the broad class of Swarm Intelligence methods. It was introduced by Kennedy and Eberhart [1] and is inspired by the social behavior of bird flocking and fish schooling. Several variations and hybrid approaches, with altered search dynamics, have been proposed, to improve the performance and the convergence characteristics of the algorithm [2]–[12]. Nevertheless, a relatively small number of PSO variants have exhibited substantial performance gains over a multitude of different real-world applications, and hence have attracted the attention of the Evolutionary Computing research community. This is strongly related to and justified by the No Free Lunch theorem (NFL) [13]. The NFL theorem mathematically proves that the average performance of any pair of optimization algorithms, across all possible optimization problems is identical, i.e. there is not an ideal algorithm for handling all possible optimization problems. The aforementioned variants successfully exploit different aspects of the PSO algorithm, either by utilizing novel velocity update rules that enhance PSO's exploratory/exploitative search power, or by incorporating special schemes to exploit the structure of the benchmark function at hand. Representative examples of the former type include fully informed topologies [10], barebones velocity updates [6], multiple swarms [3], [7], unified operators [11], [14], and comprehensive learning schemes [8]. Variants of the latter type include schemes such as cooperation and coevolution [2], [9].

A different active research direction in PSO is the integration of PSO variants in new adaptive or self-adaptive schemes, in an attempt to aggregate their characteristics and their search dynamics. To this end, Frankenstein's PSO [15] integrates three distinct algorithmic PSO components to combine their characteristics and produce an effective optimizer. Wang *et al.* [16] proposed a self-adaptive learning PSO scheme, which simultaneously adopts four PSO search strategies with different characteristics. The model adapts a probability of selecting each strategy based on their ability to generate fitter solutions during the optimization procedure. In turn, the concept of heterogeneous PSO variants has been adopted in many recent works [17]–[19]. This concept includes either sub-swarms that incorporate different meta-heuristics and cooperate with each other, or swarms with different strategies per particle which are selected from a pool of strategies in a predefined, or a dynamic manner. Finally, Unified PSO [11], [14] has been proposed as a modification of PSO that aggregates its local and global variants, combining their exploration and exploitation abilities without imposing additional requirements with respect to function evaluations.

From the above we can see that there is a large number of different PSO variants and parameter settings. Identifying which of them is most suitable for a specific problem may consume a large amount of computational cost. Thus, integrating the advantages of each variant becomes crucial. Such a scheme should be capable of choosing the most appropriate algorithm, or parameter set without wasting computational cost, i.e. learning from its environment and choosing the best candidate to apply throughout the evolution.

In this study, we borrow ideas from adaptive filter theory and statistical pattern recognition to develop an "online" algorithm adaptation technique. The proposed approach uses three state-of-the-art PSO variants, namely the Fully Informed Particle

Swarm Optimization (FIPS) [10], the Comprehensive Learning Particle Swarm Optimizer (CLPSO) [8], and the Bare Bones Particle Swarm Optimization (BBPSO) [6]. It allows each particle to randomly select amongst them to evolve in each time step. The probability of selecting each variant depends on its history of generating successful evolution movements. Extensive experimental results on 10 benchmark functions demonstrate that the proposed framework is very promising. For the majority of tested cases, the proposed framework exhibits great performance gains, while it successfully incorporates the most appropriate algorithm for the problem at hand.

The rest of the paper is organized as follows: Section II briefly describes the multinomial distribution tracker along with its main characteristics. Its incorporation into the PSO algorithm as a new strategy adaptation framework is briefly described in Section III. Section IV presents an extensive experimental and statistical analysis of the proposed framework. The paper concludes with a discussion and some pointers for future work.

II. Multinomial Distribution Tracking through exponential forgetting

In this section we briefly present the multinomial distribution and its extension to include exponential forgetting. The binomial distribution is the probability distribution of the number of "successes" in N independent Bernoulli trials, with a constant probability of success for each trial. The multinomial distribution is a generalization of the binomial distribution, in which each trial results in one out of a fixed finite number K of possible outcomes, with probabilities $\theta_1, \theta_2, \ldots, \theta_K$ and N independent trials. A random variable X_i indicates the number of times outcome i was observed over the N trials. Thus, 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:

$$P(X_1 = x_1, \ldots, X_K = x_K|\theta, N) = \frac{N!}{\prod_{i=1}^{K} x_i!} \prod_{i=1}^{K} \theta_i^{x_i}.$$

Based on a data sample D we can estimate the parameter $\widehat{\theta} = \theta(D)$ through Maximum Likelihood Estimation (MLE). The likelihood function is defined as the probability density of the data measurements given a specific value θ of the distribution parameters. Thus given a data sample D, the likelihood function is defined as: $L(\theta; D) = p(D|\theta) = p(x_1, x_2, \ldots, x_K|\theta)$. In MLE we seek the parameter $\widehat{\theta}$ that maximizes the likelihood function defined using the data sample D at hand: $L(\widehat{\theta}; D) = \max_\theta L(\theta; D)$. In the i.i.d. context the likelihood function can be written as a product of the known conditional densities for θ_i, i.e. $L(\theta; D) = p(D|\theta) = \prod_{i=1}^{K} p(x_i|\theta_i)$. Without loss of generality the maximization of the likelihood can be easily converted as a maximization of the log-likelihood or as a minimization of the negative log-likelihood function. For the multinomial distribution case one can easily calculate the MLE estimator of θ by applying Lagrange multipliers in the log-likelihood function. Hence we can obtain the MLE of the multinomial distribution by the following form: $\widehat{\theta}_i^{\mathrm{ML}} = m_k/N$, where $m_k = \sum_{i=1}^{K} x_i$.

In the context of an online strategy adaptation scheme where the optimization procedure frequently changes phases, a reasonable assumption is that the impact of each observation should be related to the time of observation. More recent information about the evolution phase is expected to be more relevant to the optimization procedure while earlier information should be slowly discarded. Here we develop a tracking framework that is based on the Recursive Least Squares (RLS) adaptive filter [20]–[24]. To this end, we incorporate weights to the likelihood function and adopt the RLS filter framework proposed in [22]–[24].

We make the assumption that the 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. As in the RLS filter, we incorporate an exponential weighting factor in the log-likelihood function and produce a new likelihood which incorporates time, $L^\lambda(\theta|D_1, D_2, \ldots, D_t)$. The new likelihood can be defined as:

$$\begin{aligned} L^\lambda(\theta|D_1, D_2, \ldots, D_t) &= \sum_{j=1}^{t} \lambda^{t-j} L(\theta|D_1, \ldots, D_j) \\ &= L(\theta|D_t) + \lambda L(\theta|D_1, \ldots, D_{t-1}), \end{aligned}$$

where $\lambda \in [0, 1]$ is a weighting factor which is also called *forgetting factor*. The forgetting factor discounts the impact of past observations on the log-likelihood and hence enables the estimated parameters to adapt to changes. As λ increases to unity all data examples are assigned equal weights, while as λ decreases more recent data samples become more important.

By applying Lagrange multipliers we obtain the MLE $\widehat{\theta}_i^{\mathrm{ML}_\lambda}$:

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

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 success of outcome i at time t. For $\lambda = 1$ the aforementioned framework corresponds to the simple case with the $\widehat{\theta}_i^{\mathrm{ML}}$ MLE.

Through this approach we can track the parameters of a multinomial distribution with the potential of forgetting the history of past observations in an exponential manner. To demonstrate the behavior of the multinomial distribution tracker with exponential forgetting we conduct a constructive simulation with an abrupt change. In particular, we construct a two class problem, where each class corresponds to the success of one strategy, with pre-specified class probabilities (here 0.8 and 0.2 respectively). After 1000 observations there is a abrupt change in which the classes (strategies) interchange their probabilities (0.2, 0.8). Hence, Figure 1 illustrates the behavior of the estimated probabilities through the multinomial distribution tracker, for four different forgetting factor values,

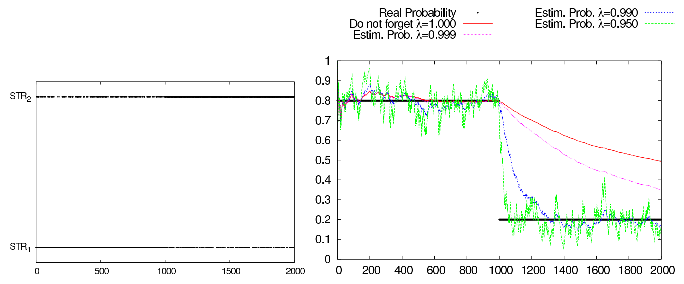

Fig. 1. Illustration of the multinomial distribution behavior in the case of an abrupt change: (left) The distribution of real successes for two different strategies. (right) The real probability against the estimated probabilities of the multinomial distribution tracker for different forgetting factor values, $\lambda \in \{0.950, 0.990, 0.999, 1.000\}$.

$\lambda \in \{0.950, 0.990, 0.999, 1\}$. We can clearly observe that in the case of an abrupt change point, a lower forgetting factor enables a faster adaptation. In contrast the no forgetting case ($\lambda = 1$) performs very poorly, and cannot successfully track the true parameters after the change.

III. The Multinomial distribution-based Particle Swarm Optimization framework

In this section, we discuss the main concepts behind the proposed framework, namely the Multinomial distribution-based PSO (MultiPSO). The PSO algorithm is a population–based stochastic algorithm that exploits a population of individuals to effectively probe promising regions of the search space. Therefore, each individual (*particle*) of the population (*swarm*) moves with an adaptable velocity within the search space and retains in its memory the best position it ever encountered. A thorough description of the PSO algorithm can be found in [4], [6], [11], [25], [26].

The proposed framework introduces two main concepts different from the standard PSO. Firstly, it probabilistically assigns to each particle one PSO variant, k_{str}, chosen from a pool of candidate algorithms, $k_{str} \in \{1, 2, \ldots, K\}$. Consequently, based on the particle's movements, it adopts its probability during the evolution stages through the aforementioned multinomial distribution tracker. The remaining evolution steps of the PSO algorithm remain the same. More specifically, we incorporate a pool of K state-of-the-art PSO variants. Here, we utilize three well-known and widely used variants that have efficiently tackled several real or artificial problem landscapes [6], [8], [10], namely the FIPS, the CLPSO, and the BBPSO algorithms. It is obvious that any PSO variant could be incorporated into the pool to enhance the exploratory and exploitative power of the proposed framework.

In turn, one of the available algorithms is assigned to each particle based on a probability. This probability is adapted at each generation through the aforementioned multinomial distribution tracker, based on its successful and unsuccessful evolution steps. An evolution step is characterized as successful, if the applied algorithm succeeds to find a better personal best position of the particle.

In detail, for each algorithm i in the pool $i \in \{1, 2, \ldots, K\}$, we incorporate a memory for its successful (SStep(i)) and unsuccessful (FStep(i)) evolution steps. Thus, for each algorithm we assign a score based on the percentage of its successful evolution steps during the last generation. The score, Score(i), can be calculated through the following formula:

Algorithm 1 The MultiPSO algorithmic scheme

```
1:  Initialize particles in the swarm
2:  Initialize the multinomial distribution tracker, for each algorithm
    $i : n_i(t_0) = 0$ and $\widehat{\theta}_i^{\mathrm{ML}_\lambda}(t_0) = \frac{1}{K}$.
3:  **for** each time step $t$ **do**
4:      **for** each particle $j$ in the swarm **do**
5:          **Sample** $k_{str}$ **from the multinomial distribution with**
            **parameters** $\widehat{\theta}_i^{\mathrm{ML}_\lambda}(t)$.
6:          **Update its position** using algorithm $k_{str}$, $k_{str} \in$
            $\{$1) FIPS, 2) CLPSO, 3) BBPSO$\}$.
7:          **Evaluate particle** $j$
8:          **Update social and cognitive experience**
9:          **if** particle $j$ has a better personal best position **then**
10:             Add a success point to the corresponding strategy
11:         **else**
12:             Add a failure point to the corresponding strategy
13:         **end if**
14:     **end for**
15:     **Update the multinomial distribution tracker through**
        **Eqs. (2)–(3)**
16: **end for**
```

$$\mathrm{Score}(i) = \mathrm{round}\left(100 \frac{\mathrm{w}(i)}{\sum_{i=1}^{K} \mathrm{w}(i)}\right), \text{with}$$

$$\mathrm{w}(i) = \left(\frac{\mathrm{SStep(i)}}{\mathrm{SStep(i)} + \mathrm{FStep(i)} + \varepsilon} + p_{\min}\right), \qquad (3)$$

where $p_{\min} = 0.01$ is a small constant that prevents the extinction of an algorithm, while $\varepsilon = 10^{-6}$ helps to avoid divisions by zero, in the case that an algorithm has not been selected in the previous generation. The rounding procedure as well as the multiplication by 100 will fix the score to the required integer value, by the multinomial distribution. The final score assists the algorithm which produces the higher percentage of successful evolution steps in the last generation. To this end, the multinomial distribution tracker learns from the current evolution stage and promotes the algorithm that is more likely to efficiently evolve the swarm 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.

IV. Experimental results

In this section we perform an experimental evaluation of the proposed approach. We employ ten high dimensional, scalable benchmark functions with shifted search space domains and different characteristics. The first six functions have been acquired from the recent CEC'2008 Special Session on Large Scale Global Optimization [27]. Two of them are unimodal functions (f_1 and f_2), while the next four are multimodal with a large number of local optima ($f_3 - f_6$). The remaining four test functions are hybrid composition functions, recently proposed in [28], and correspond to the functions $f_{16} - f_{19}$ of the test suite. In this study, we consider the 50-dimensional versions of the aforementioned benchmark functions. A detailed description of the benchmark set can be found in [27], [28].

To demonstrate the efficiency of the proposed approach, we compare it with five state-of-the-art PSO variants, namely the local version of the PSO with constriction factor (xPSOl), the local version of the inertia weight PSO with a linearly decreasing weight rule (wPSOl), the Bare Bones Particle Swarm Optimization (BBPSO) [6], the Fully Informed Particle Swarm Optimization (FIPS) [10], and the Comprehensive Learning Particle Swarm Optimizer (CLPSO) [8]. We implement the proposed approach with three different predefined forgetting factor values, $\lambda \in \{0.92, 0.99, 1\}$. 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 $w \approx 12.5$, or $w \approx 100$ generations respectively. The sliding window can be approximated using the λ parameter, through: $w \approx 1/(1 - \lambda)$ [21]. In turn, $\lambda = 1$ corresponds to no forgetting. A brief analysis on the forgetting factor parameter is demonstrated bellow (see Section IV-B).

For each simulation and method, we have initialized the swarms using a uniform random number generator with the same random seeds. Furthermore, all methods have been implemented with the default parameters settings as have been proposed in the literature. Regarding the PSO with constriction factor parameters, the common setting of $\varphi = 4.1$, $\chi = 0.72984$ and $c_1 = c_2 = 2.05$ and a ring topology has been utilized [4], [29]. The swarm size has been kept fixed to $NP = 100$ particles and for each simulation, a budget of $\max NFEs = 5000 \cdot D$ function evaluations has been employed [27].

To evaluate the performance of the PSO variants we will use the *solution error measure*, or simply *error*, defined as $f(x') - f(x^\star)$, where $x^\star$ is the global optimum of the benchmark function and x' is the best solution achieved within a budget of $\max NFEs$ function evaluations. Each algorithm was executed independently 50 times to obtain an estimation of the median (*Median*), the mean solution error (*Mean*), its standard deviation (*St.D.*) and the success (*Success*). For each benchmark function we use boldface font to indicate the best performing algorithm in terms of median and mean solution error. To evaluate the statistical significance of the observed performance differences, for each variant we conducted three two-sided Wilcoxon rank sum tests between the corresponding variant and MultiPSO, using the aforementioned λ values, $\lambda \in \{0.92, 0.99, 1\}$. The null hypothesis in each test is that the samples compared are independent samples from identical continuous distributions with equal medians. We mark with "+" the cases when the null hypothesis is rejected at the 5% significance level and the proposed approach exhibits superior performance. Mark "–" indicates that the null hypothesis is rejected at the same level of significance and the proposed approach exhibits inferior performance, while mark "=" indicates that the performance difference is not statistically significant.

To this end, Table I reports the experimental results on the 50–dimensional versions of the considered benchmark set. It is evident that the evolution of the state-of-the-art PSO variants through the multinomial distribution tracker results in an enhanced PSO scheme with superior performance. Generally speaking, MultiPSO exhibits a significant performance improvement, in terms of median and mean error values, in the majority of the tested functions. More specifically, in seven out of ten functions, ($f_1 - f_3$ and $f_5 - f_8$), there is a MultiPSO variant that exhibits a significantly better performance against all state-of-the-art PSO variants. In three cases, (f_1, f_5, and f_6), all MultiPSO versions exhibit 100% success. Only in three functions (f_4, f_9 and f_{10}) either CLPSO or FIPS, exhibit superior performance against all MultiPSO versions. Finally, in f_8, FIPS exhibits a significantly better performance against MultiPSO$_{\lambda=0.92}$, MultiPSO$_{\lambda=0.99}$ and a significantly worse performance against MultiPSO$_{\lambda=1}$.

Regarding the three MultiPSO versions, the results in the ten considered benchmark functions do not allow us to draw safe conclusions. In four out of ten functions, no statistically significant differences exist between the three approaches (f_1, f_3, f_5, and f_6). On the remaining functions, there are three functions in which MultiPSO with a forgetting strategy, i.e. $\lambda \in \{0.92, 0.99\}$, exhibit significant performance gains (f_4, f_7, f_{10}), while in the f_8 and f_9 functions MultiPSO$_{\lambda=1}$ performs better. More specifically, the forgetting mechanism does not seem to enhance the proposed approach in the f_8 and f_9 functions, since MultiPSO$_{\lambda=1}$ exhibits significant performance differences against both MultiPSO$_{\lambda=0.92}$ and MultiPSO$_{\lambda=0.99}$. Similarly, in the f_2 function, MultiPSO$_{\lambda=1}$ is significantly better than the MultiPSO$_{\lambda=0.99}$, while it performs equally well with MultiPSO$_{\lambda=0.92}$. Among the MultiPSO approaches that incorporate the forgetting mechanism, MultiPSO$_{\lambda=0.92}$ performs better, since it exhibits a significantly better performance in the f_4 and f_{10} functions against both MultiPSO$_{\lambda=0.99}$ and MultiPSO$_{\lambda=1}$. Additionally MultiPSO$_{\lambda=0.92}$ is better against the MultiPSO$_{\lambda=1}$ in the f_7 function.

A first suggestion that can be made from the aforementioned observations is that the forgetting procedure exhibits a problem dependent behavior. This behavior seems reasonable since the forgetting mechanism is applied in the probability adaptation of each strategy, i.e. at each evolution stage an algorithm may perform better and needs to forget either slowly or quickly its history based on the current problem's surface. Nevertheless, there are relatively few cases where the forgetting approaches exhibit significant performance deterioration against the no-forgetting approach.

To illustrate the behavior of the proposed approach during the simulations, in Figure 2, we provide convergence graphs for the first six shifted functions ($f_1 - f_6$). The graphs demonstrate median solution error value curves of 50 independent simulations for all PSO variants considered in this paper. As expected, the graphs capture the previously observed behavior of the PSO algorithms, while they indicate that in most of the cases the MultiPSO approach either enhances the convergence

TABLE I

ERROR VALUES OF THE PROPOSED PSO METHOD (MULTIPSO) AND FIVE STATE-OF-THE-ART PSO VARIANTS ON THE 50–DIMENSIONAL VERSIONS OF THE CONSIDERED BENCHMARK SET.

Algorithm	*Median*	*Mean*	*St.D.*	NFE	*Success*	*St. Sig.*	*Median*	*Mean*	*St.D.*	NFE	*Success*	*St. Sig.*
	f_1 : Shifted Sphere Function						f_2 : Shifted Schwefel's Problem 2.21					
xPSOI	7.405e+03	7.508e+03	9.342e+02	N/A	0.0	(+/+/+)	2.911e+01	2.915e+01	1.621e+00	N/A	0.0	(+/+/+)
wPSOI	2.519e+03	2.583e+03	4.175e+02	N/A	0.0	(+/+/+)	2.588e+01	2.564e+01	2.326e+00	N/A	0.0	(+/+/+)
BBPSO	0.000e+00	0.000e+00	0.000e+00	N/A	0.0	(+/+/+)	5.372e+01	5.245e+01	4.420e+00	N/A	0.0	(+/+/+)
CLPSO	2.550e-02	2.556e-02	5.718e-03	N/A	0.0	(+/+/+)	5.442e+01	5.400e+01	2.992e+00	N/A	0.0	(+/+/+)
FIPS	6.746e+03	6.836e+03	7.820e+02	N/A	0.0	(+/+/+)	3.478e+01	3.462e+01	2.300e+00	N/A	0.0	(+/+/+)
MultiPSO$_{\lambda=0.92}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.600e+05	100.0	(=/=/=)	**9.945e-01**	**1.216e+00**	5.522e-01	N/A	0.0	(=/=/=)
MultiPSO$_{\lambda=0.99}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.598e+05	100.0	(=/=/=)	1.141e+00	1.200e+00	4.202e-01	N/A	0.0	(=/=/–)
MultiPSO$_{\lambda=1.00}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.600e+05	100.0	(=/=/=)	1.032e+00	1.011e+00	2.798e-01	N/A	0.0	(=/+/=)
	f_3 : Shifted Rosenbrock's Function						f_4 : Shifted Rastrigin's Function					
xPSOI	3.314e+06	3.838e+06	3.056e+06	N/A	0.0	(+/+/+)	2.335e+02	2.319e+02	2.580e+01	N/A	0.0	(+/+/+)
wPSOI	1.782e+02	1.876e+02	4.294e+01	N/A	0.0	(+/+/+)	1.703e+02	1.678e+02	1.957e+01	N/A	0.0	(+/+/+)
BBPSO	2.952e+02	3.141e+02	1.056e+02	N/A	0.0	(+/+/+)	1.483e+02	1.489e+02	2.166e+01	N/A	0.0	(+/+/+)
CLPSO	9.838e+02	9.988e+02	1.944e+02	N/A	0.0	(+/+/+)	**3.305e-01**	**3.407e-01**	8.753e-02	N/A	0.0	(–/–/–)
FIPS	9.503e+05	2.023e+06	2.493e+06	N/A	0.0	(+/+/+)	1.846e+02	1.851e+02	1.531e+01	N/A	0.0	(+/+/+)
MultiPSO$_{\lambda=0.92}$	4.565e+01	4.709e+01	7.751e+00	N/A	0.0	(=/=/=)	1.686e+01	1.791e+01	7.200e+00	N/A	0.0	(=/+/+)
MultiPSO$_{\lambda=0.99}$	4.554e+01	4.739e+01	1.152e+01	N/A	0.0	(=/=/=)	3.480e+01	3.340e+01	6.900e+00	N/A	0.0	(–/=/+)
MultiPSO$_{\lambda=1.00}$	**4.541e+01**	**4.576e+01**	7.768e+00	N/A	0.0	(=/=/=)	5.804e+01	5.740e+01	1.255e+01	N/A	0.0	(–/–/=)
	f_5 : Shifted Griewank's Function						f_6 : Shifted Ackley's Function					
xPSOI	4.189e+01	4.286e+01	6.178e+00	N/A	0.0	(+/+/+)	8.428e+00	8.476e+00	5.340e-01	N/A	0.0	(+/+/+)
wPSOI	1.316e+01	1.325e+01	2.603e+00	N/A	0.0	(+/+/+)	2.840e+00	2.867e+00	4.881e-01	N/A	0.0	(+/+/+)
BBPSO	0.000e+00	3.600e-04	1.687e-03	N/A	0.0	(+/+/+)	1.438e+00	1.045e+00	8.457e-01	N/A	0.0	(+/+/+)
CLPSO	3.400e-02	3.500e-02	8.899e-03	N/A	0.0	(+/+/+)	5.900e-02	6.018e-02	1.020e-02	N/A	0.0	(+/+/+)
FIPS	4.009e+01	4.027e+01	6.103e+00	N/A	0.0	(+/+/+)	8.204e+00	8.091e+00	6.703e-01	N/A	0.0	(+/+/+)
MultiPSO$_{\lambda=0.92}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.657e+05	100.0	(=/=/=)	**0.000e+00**	**0.000e+00**	0.000e+00	2.041e+05	100.0	(=/=/=)
MultiPSO$_{\lambda=0.99}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.677e+05	100.0	(=/=/=)	**0.000e+00**	**0.000e+00**	0.000e+00	2.046e+05	100.0	(=/=/=)
MultiPSO$_{\lambda=1.00}$	**0.000e+00**	**0.000e+00**	0.000e+00	1.646e+05	100.0	(=/=/=)	**0.000e+00**	**0.000e+00**	0.000e+00	2.053e+05	100.0	(=/=/=)
	f_7 : Hybrid Composition Function 1 (f_{16} [28])						f_8 : Hybrid Composition Function 2 (f_{17} [28])					
xPSOI	1.937e+03	1.960e+03	3.495e+02	N/A	0.0	(+/+/+)	1.743e+02	1.706e+02	3.124e+01	N/A	0.0	(+/+/+)
wPSOI	7.098e+02	6.993e+02	1.382e+02	N/A	0.0	(+/+/+)	1.701e+02	1.677e+02	2.393e+01	N/A	0.0	(+/+/+)
BBPSO	4.285e+01	4.197e+01	1.423e+01	N/A	0.0	(+/+/+)	1.571e+02	1.568e+02	2.567e+01	N/A	0.0	(+/+/+)
CLPSO	6.576e+00	6.646e+00	6.620e-01	N/A	0.0	(+/+/+)	1.473e+02	1.472e+02	1.259e+01	N/A	0.0	(+/+/+)
FIPS	1.686e+03	1.677e+03	2.537e+02	N/A	0.0	(+/+/+)	1.281e+01	2.450e+01	2.547e+01	N/A	0.0	(–/–/+)
MultiPSO$_{\lambda=0.92}$	9.215e-05	1.136e-04	9.575e-05	N/A	0.0	(=/=/+)	5.980e+01	6.549e+01	2.812e+01	N/A	0.0	(=/–/–)
MultiPSO$_{\lambda=0.99}$	**9.147e-05**	**1.100e-04**	5.146e-05	N/A	0.0	(=/=/+)	4.354e+01	4.952e+01	2.597e+01	N/A	0.0	(+/=/–)
MultiPSO$_{\lambda=1.00}$	1.260e-04	1.406e-04	6.537e-05	N/A	0.0	(–/–/=)	**1.261e+01**	**1.753e+01**	1.056e+01	N/A	0.0	(+/+/=)
	f_9 : Hybrid Composition Function 3 (f_{18} [28])						f_{10} Hybrid Composition Function 4 (f_{19} [28])					
xPSOI	6.600e+01	6.504e+01	4.219e+00	N/A	0.0	(+/+/+)	2.085e-12	2.643e-12	2.696e-12	1.928e+05	100.0	(+/+/+)
wPSOI	5.711e+01	5.654e+01	4.577e+00	N/A	0.0	(+/+/+)	1.568e-07	1.799e-07	1.231e-07	N/A	0.0	(+/+/+)
BBPSO	5.352e+01	5.247e+01	6.974e+00	N/A	0.0	(+/+/+)	9.474e-05	2.999e-04	8.665e-04	N/A	0.0	(+/+/+)
CLPSO	**4.168e+00**	**4.212e+00**	3.270e-01	N/A	0.0	(–/–/–)	6.998e-03	7.215e-03	1.226e-03	N/A	0.0	(+/+/+)
FIPS	3.824e+01	3.784e+01	6.513e+00	N/A	0.0	(+/+/+)	**7.652e-22**	**1.090e-21**	1.267e-21	1.267e+05	100.0	(–/–/–)
MultiPSO$_{\lambda=0.92}$	1.764e+01	1.814e+01	4.414e+00	N/A	0.0	(=/=/–)	1.935e-17	2.280e-17	1.520e-17	1.749e+05	100.0	(=/+/+)
MultiPSO$_{\lambda=0.99}$	1.742e+01	1.789e+01	3.898e+00	N/A	0.0	(=/=/–)	2.968e-17	3.631e-17	2.208e-17	1.763e+05	100.0	(–/=/+)
MultiPSO$_{\lambda=1.00}$	1.319e+01	1.320e+01	2.246e+00	N/A	0.0	(+/+/=)	4.761e-16	1.504e-15	2.336e-15	1.795e+05	100.0	(–/–/=)

of a PSO variant or converges similarly. There are relatively few cases where the proposed approaches exhibit performance deterioration.

Additionally, Figure 3 illustrates a typical behavior of the probabilities adapted by the multinomial tracker over the first six shifted functions ($f_1 - f_6$). The graphs demonstrate median probability value curves of 50 independent simulations for the considered lambda values. It can be easily observed that the adaptation of the strategy probabilities behave differently based on the benchmark problem at hand as well as the evolution phase, in which there are numerous probability trends. More specifically, it can be easily observed that in the majority of the illustrated cases, there are three main stages through the evolution procedure, approximately every 1000 generations. In the first evolution stage, there are rapid changes in the multinomial probabilities ($f_1 - f_6$), with many alterations for the leading probability position. A similar behavior is also exhibited in the last stage (f_1, f_4, f_5), while in the middle stage the probability curves exhibit stable and robust trends. In general, this is an expected behavior, since each problem exhibits different characteristics and the strategies perform differently. Moreover, concerning the forgetting factor values, it can be observed that when the multinomial tracker does not forget ($\lambda = 1$), the probability curves are stable, with no rapid changes throughout the evolution. On the contrary, the more the multinomial tracker forgets (i.e. the λ value decreases) the more rapidly the probabilities change. This trend can be easily observed for the $\lambda = 0.99$ and $\lambda = 0.92$ cases. It is worth noting that although in three cases (f_4, f_9, and f_{10}) the proposed framework does not exhibit superior performance, the multinomial distribution tracker tries to capture the best performing algorithm, e.g. in Figure 3 observe the trends of the

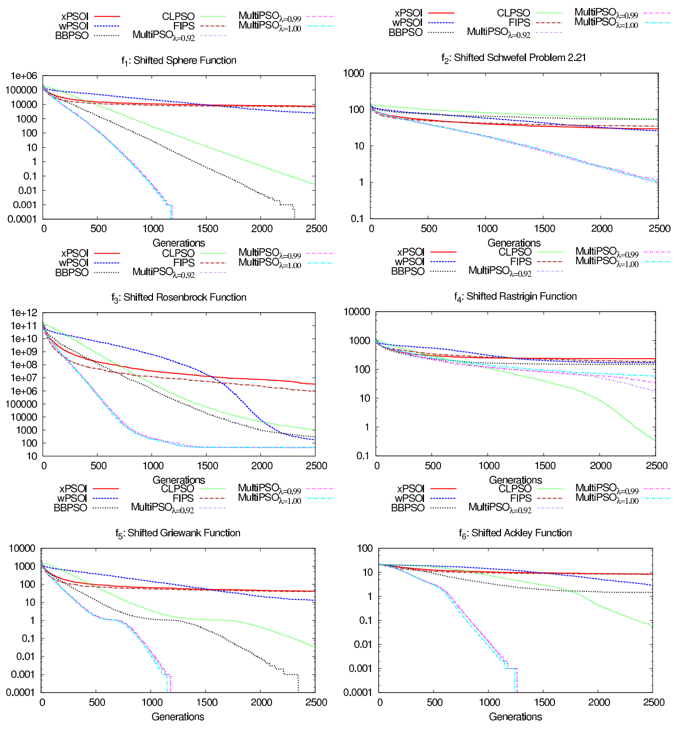

Fig. 2. Convergence graph (median curves) for the considered PSO variants over the shifted 50–dimensional functions $f_1 - f_6$. The horizontal axis illustrates the number of generations, and the vertical axis illustrates the median of solution error values over 50 independent simulations.

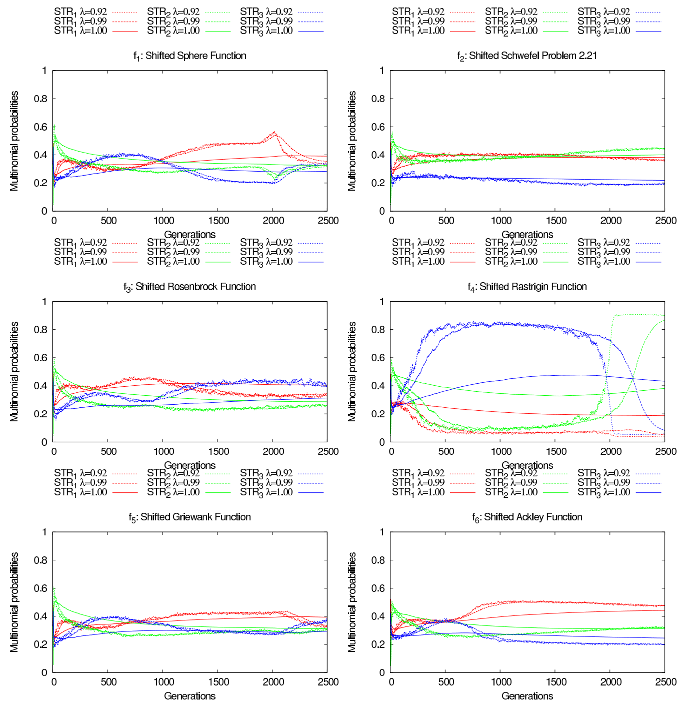

Fig. 3. Multinomial probabilities graph (median curves) for the lambda parameter ($\lambda \in \{0.92, 0.99, 1\}$) over the shifted 50–dimensional functions $f_1 - f_6$. The horizontal axis illustrates the number of generations, and the vertical axis illustrates the median probabilities per strategy over 50 independent simulations.

TABLE II
AVERAGE RANKINGS OF THE CONSIDERED PSO VARIANTS, ACHIEVED BY THE FRIEDMAN AND THE QUADE STATISTICAL TESTS

	Average Ranking			
	Friedman test		Quade test	
Algorithms	Score	Rank	Score	Rank
xPSOl	7.3999	(8)	7.1090	(8)
wPSOl	5.9000	(7)	6.0000	(7)
BBPSO	5.1000	(5)	5.3636	(6)
CLPSO	4.7000	(4)	5.0000	(5)
FIPS	5.5000	(6)	4.9818	(4)
MultiPSO$_{\lambda=0.92}$	2.5000	(2)	2.3818	(1)
MultiPSO$_{\lambda=0.99}$	2.5000	(2)	2.6181	(3)
MultiPSO$_{\lambda=1.00}$	2.4000	(1)	2.5454	(2)
Statistic	40.30000		7.40772	
p-value	1.1028e-6		1.7568e-6	

estimated probabilities for the f_4 function, where in the first and last stages of evolution the scheme promotes the CLPSO variant, which is the best performing algorithm.

The aforementioned observations suggest the need to adapt the forgetting factor value through different evolution phases by utilizing either an adaptive or a self-adaptive procedure. This is a very interesting research area that we intend to extensively study in the future.

A. *Statistical Significance Analysis*

To evaluate the statistical significance of the observed performance differences, we apply the Friedman and the Quade tests [30], [31]. These tests rank the performance of a set of k algorithms and can detect if there exists a significant difference in the performance of at least two algorithms. Table II, depicts the average rankings computed through the aforementioned tests, while at the bottom of the table we illustrate the statistics of each test along with its corresponding p-values.

The p-values computed through the Friedman and the Quade statistical tests (1.1028e-6 and 1.7568e-6, respectively) along with the Iman and Davenport extension ($F_f = 12.2121$, p-value: 9.5316e-10), strongly suggest the existence of significant differences among the considered algorithms, at the $\alpha = 0.05$ level of significance. Notice that the Quade test is a variation of the Friedman test, which takes into account the fact that some cases in the sample maybe more important than others. Thus, it calculates scaled rankings depending on the differences observed in the samples [31]. The Quade test may therefore indicate that the corresponding algorithms exhibit significant differences in more benchmark functions at any given level of significance.

It can be observed that all versions of the proposed approach (MultiPSO$_{\lambda=0.92}$, MultiPSO$_{\lambda=0.99}$, and MultiPSO$_{\lambda=1}$, always come in the first three positions of the rankings, while the next three positions, are usually occupied by BBPSO, CLPSO, and FIPS. This observation suggests that combining the dynamics and the characteristics of the aforementioned three PSO variants through the proposed methodology, is an efficient approach with a lot of potential. Moreover, regarding the MultiPSO versions with different forgetting factor values,

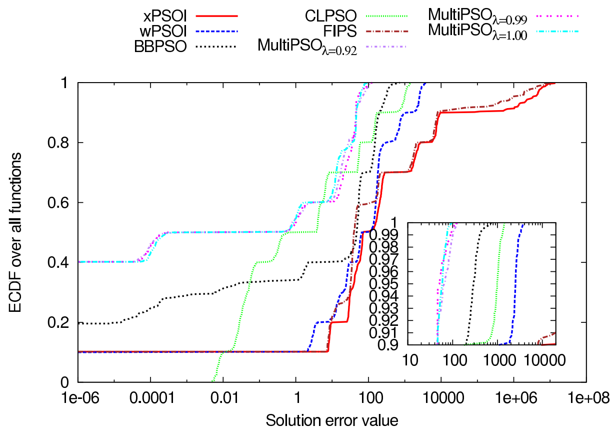

Fig. 4. Empirical cumulative probability distribution of the solution error values for the proposed PSO approach against five state-of-the-art PSO variants on the 50–dimensional versions of the considered benchmark functions.

the Friedman test suggests that MultiPSO$_{\lambda=1}$ comes first, i.e. a no forgetting procedure exhibits better behavior in the considered benchmark functions. On the other hand, the Quade test indicates that in the most benchmark functions a rapidly forgetting procedure ($\lambda = 0.92$) is suitable.

We conclude the statistical analysis with the application of the Empirical Cumulative probability Distribution Function of the performance error values ($ECDF$). The $ECDF$ graphically demonstrates the performance of the implemented algorithms on all benchmark functions and can be utilized as an overall performance visualization statistic. Specifically, for an algorithm A on a function f, the error value (error) achieved by A on function f is computed. Therefore, smaller values of error correspond to better performance. The $ECDF$ of errors for an algorithm in a number of functions n_f is a cumulative probability distribution function defined as: $ECDF(x) = (1/n_f)\sum_{i=1}^{n_f} I(\text{error}_i \leqslant x)$, where $I(\cdot)$ is the indicator function. In other words, the $ECDF$ measure captures the empirical probability of observing an *error* value smaller or equal to x. First, we compute the *errors* for all considered algorithms on all the functions and then we compute the $ECDF$ for each algorithm. This enables a summarizing comparison of the algorithms in all the benchmarks, as larger $ECDF$ values for the same argument correspond to better performance.

Figure 4, illustrates the $ECDF$ of the error for all versions of the proposed MultiPSO versus the five state-of-the-art PSO variants. It can be clearly observed that all proposed approaches exhibit a great potential on the considered function set. All MultiPSO versions exhibit higher ECDF values compared with the other variants in the majority of the observed error values. Only CLPSO exhibits higher ECDF values for some observations. In general, the MultiPSO versions produce one or more orders of magnitude lower error values, i.e. the MultiPSO curves reach unity at approximately error ≈ 100, while BBPSO and CLPSO curves at error ≈ 1000 (please refer to the zoomed sub-figure inside Figure 4). A similar behavior can be observed for the FIPS and xPSOl variants, where they

TABLE III
AVERAGE RANKINGS OF THE MULTIPSO APPROACH FOR SEVERAL FORGETTING FACTOR VALUES, ACHIEVED BY THE FRIEDMAN, AND QUADE STATISTICAL TESTS

Forgetting with		Average Ranking	
lambda value	sliding window	Friedman test	Quade test
$\lambda = 0.000$	immediately forget	10.2500 (14)	9.2636 (9)
$\lambda = 0.500$	$w = 1.0000$	9.0500 (4)	**8.5181** (3)
$\lambda = 0.800$	$w = 5.0000$	9.3500 (10)	8.7181 (5)
$\lambda = 0.900$	$w = 10.0000$	9.1500 (6)	9.1181 (6)
$\lambda = 0.910$	$w = 11.1100$	**8.3500** (2)	8.6272 (4)
$\lambda = 0.920$	$w = 12.5000$	**7.0500** (1)	**7.3000** (1)
$\lambda = 0.930$	$w = 14.2857$	9.3500 (10)	9.3181 (10)
$\lambda = 0.940$	$w = 16.6667$	9.1500 (6)	9.1909 (8)
$\lambda = 0.950$	$w = 20.0000$	**8.6499** (3)	**8.1181** (2)
$\lambda = 0.960$	$w = 25.0000$	9.9500 (13)	9.4090 (11)
$\lambda = 0.970$	$w = 33.3333$	9.1500 (6)	9.1727 (7)
$\lambda = 0.980$	$w = 50.0000$	10.6499 (15)	10.2636 (14)
$\lambda = 0.990$	$w = 100.0000$	11.1500 (17)	10.9181 (16)
$\lambda = 0.995$	$w = 200.0000$	11.3500 (18)	11.4272 (18)
$\lambda = 0.996$	$w = 250.0000$	9.2500 (9)	9.9181 (12)
$\lambda = 0.998$	$w = 500.0000$	10.6500 (16)	11.3727 (17)
$\lambda = 0.999$	$w = 1000.0000$	9.4500 (12)	10.2818 (15)
$\lambda = 1.000$	Do not forget	9.0500 (4)	10.0636 (13)
	Statistic	6.48596	0.37953
	p-value	0.9892	0.9879

reach unity at approximately error $\approx 10^7$. Among the three MultiPSO versions there is no visible difference. Only in a very small range of error values, MultiPSO$_{\lambda=1}$ exhibits higher ECDF values.

B. A brief analysis on the forgeting factor parameter

In this subsection, we briefly analyze the forgetting factor parameter and try to capture its main characteristics and understand its behavior. To this end, we have utilized eighteen different forgetting factor values in the range $[0, 1]$. For each forgetting factor value we have conducted 50 independent simulations over all considered benchmark functions and ranked their performance through the Friedman and the Quade statistical ranking tests. Table III demonstrates the average rankings computed by the tests. For each forgetting factor value, we report the corresponding sliding window size and its ranking scores. Notice that the forgetting factor corresponds to an exponential function in terms of a sliding window size. Additionally, $\lambda = 0$ corresponds to an immediately forgetting procedure, while $\lambda = 1$ corresponds to a non-forgetting procedure.

The p-values computed through the statistical tests (0.9892 and 0.9879, respectively) suggest that the performance among the considered forgetting factors does not exhibit statistically significant differences. The considered statistical tests take into account the general behavior of the algorithms in all test cases. We observe that in the specific benchmark suite, we cannot draw safe conclusions regarding the forgetting procedure. Nevertheless, given their previously documented performance, we can conclude that the majority of the MultiPSO$_\lambda$ approaches outperform the state-of-the-art PSO variants. Moreover, it can be observed that there exists a forgetting factor value that exhibits better performance than the non-forgetting and the immediately forgetting cases. To clearly show the three better performing forgetting values, we mark them with a bold-face font. To conclude, the overall most promising forgetting factor value seems to be in the vicinity of $\lambda = 0.92$.

V. Conclusions

Recent developments in the PSO research community suggest that the advantages of several PSO variants can be exploited by integrating them in adaptive schemes. We attempt to exploit the characteristics of different PSO variants so as to improve their performance. Borrowing ideas from adaptive filter theory we develop an "online" algorithm 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 strategies involved.

Extensive 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 five state-of-the-art PSO variants. The multinomial distribution tracker is able to successfully capture the most appropriate algorithm for the problem at hand.

The most appropriate degree of forgetting depends on the evolution stage, as well as the problem. It would be interesting to further study it in the future, and develop an adaptive forgetting factor scheme. We also intent to utilize different PSO variants and study their performance on more benchmark functions.

Acknowledgment

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] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *In Proceedings 6th Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.

[2] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, june 2004.

[3] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[4] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[5] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Evolving cognitive and social experience in particle swarm optimization through differential evolution," in *IEEE Congress on Evolutionary Computation, 2010, (CEC 2010)*, 2010, pp. 1–8.

[6] J. Kennedy, "Bare bones particle swarms," in *IEEE Swarm Intelligence Symposium, 2003. SIS '03.*, 2003, pp. 80–87.

[7] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proceedings of the 2005 IEEE Swarm Intelligence Symposium, 2005. SIS '05.*, June 2005, pp. 124–129.

[8] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, June 2006.

[9] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, p. to appear, 2011, DOI:10.1109/TEVC.2011.2112662.

[10] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, June 2004.

[11] K. E. Parsopoulos and M. N. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*. Hershey, PA, U.S.A.: Information Science Reference (IGI Global), 2010.

[12] L. M. Hiot, Y. S. Ong, B. K. Panigrahi, Y. Shi, and M.-H. Lim, Eds., *Handbook of Swarm Intelligence*, ser. Adaptation, Learning, and Optimization. Springer Berlin Heidelberg, 2010, vol. 8.

[13] D. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[14] K. E. Parsopoulos and M. N. Vrahatis, "UPSO: A unified particle swarm optimization scheme," in *Lecture Series on Computer and Computational Sciences, Proceedings of the International Conference of "Computational Methods in Sciences and Engineering" (ICCMSE 2004)*, vol. 1, 2004, pp. 868–873.

[15] M. A. M. de Oca, T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's pso: A composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.

[16] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4515–4538, 2011.

[17] O. Olorunda and A. P. Engelbrecht, "An analysis of heterogeneous cooperative algorithms," in *IEEE Congress on Evolutionary Computation, 2009. (CEC 2009)*, 2009, pp. 1562–1569.

[18] M. A. Montes de Oca, J. Peña, T. Stützle, C. Pinciroli, and M. Dorigo, "Heterogeneous particle swarm optimizers," in *IEEE Congress on Evolutionary Computation, 2009, (CEC 2009)*, P. Haddow *et al.*, Eds. Piscataway, NJ: IEEE Press, 2009, pp. 698–705.

[19] A. P. Engelbrecht, "Heterogeneous particle swarm optimization," in *Proceedings of the 7th international conference on Swarm intelligence*, ser. ANTS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 191–202.

[20] S. Haykin, *Adaptive filter theory (3rd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[21] M. Niedzwiecki, *Identification of Time-varying Processes*. New York: John Wiley & Sons, 2000.

[22] N. G. Pavlidis, D. K. Tasoulis, N. M. Adams, and D. J. Hand, "λ-perceptron: An adaptive classifier for data streams," *Pattern Recognition*, vol. 44, no. 1, pp. 78–96, 2011.

[23] ——, "Adaptive consumer credit classification," *Journal of the Operational Research Society*, vol. to appear, 2012.

[24] C. Anagnostopoulos, "A statistical framework for streaming data analysis," Ph.D. dissertation, Imperial College London, UK, January 2010.

[25] R. Eberhart, P. Simpson, and R. Dobbins, *Computational intelligence PC tools*. San Diego, USA: Academic Press Professional, Inc., 1996.

[26] M. Clerc, *Particle Swarm Optimization*. ISTE Publishing Company, 2006.

[27] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, "Benchmark functions for the CEC'2008 special session and competition on large scale global optimization," Nature Inspired Computation and Applications Laboratory, China, Tech. Rep., 2007.

[28] M. Lozano, D. Molina, and F. Herrera, "Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 15, no. 11, pp. 2085–2087, 2011.

[29] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *IEEE Swarm Intelligence Symposium, 2007. SIS 2007.*, 2007, pp. 120–127.

[30] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, March 2011.

[31] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, May 2010.