Particle Swarm Optimization in the EDAs Framework

Chapter · April 2011

DOI: 10.1007/978-3-642-20505-7_7

2 authors:

Valentino Santucci
Università degli Studi di Perugia
15 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Alfredo Milani
Università degli Studi di Perugia
46 PUBLICATIONS **174** CITATIONS
SEE PROFILE

Particle Swarm Optimization in the EDAs framework

Valentino Santucci, Alfredo Milani

Abstract Particle Swarm Optimization (PSO) is a popular optimization technique based on swarm intelligence concepts. Estimation of Distribution Algorithms (EDAs) are a relatively new class of evolutionary algorithms which build a probabilistic model of the population dynamics and use this model to sample new individuals. Recently, the hybridization of PSO and EDAs is emerged as a new research trend. In this paper, we introduce a new hybrid approach that uses a mixture of Gaussian distributions. The obtained algorithm, called PSEDA, can be seen as an implementation of the PSO behaviour in the EDAs framework. Experiments on well known benchmark functions have been held and the performances of PSEDA are compared with those of classical PSO.

1 Introduction and Related Work

Particle swarm optimization (PSO) [1] is a population-based stochastic approach mainly used for solving continuous optimization problems (although some attempts to handle combinatorial problems have been proposed [2]). Inspired by the behavioural model of bird flocking, a population of particles (the candidate solutions) move in the search space at the quest of the optimum point.

Estimation of Distribution Algorithms (EDAs) [3] are a class of evolutionary algorithms (EAs) where a population of candidate solutions is sampled from a probabilistic model learned from the previous generations. Hence, conversely from classical EAs, EDAs does not use variation operators (like crossover or mutation), but

Valentino Santucci
Department of Mathematics and Computer Science, University of Perugia, via Vanvitelli 1, 06123 Perugia (Italy), e-mail: valentino.santucci@dmi.unipg.it

Alfredo Milani
Department of Mathematics and Computer Science, University of Perugia, via Vanvitelli 1, 06123 Perugia (Italy), e-mail: milani@unipg.it

rely on purely probabilistic methods. Although mainly used for discrete problems, some continuous applications of EDAs have been proposed [4].

Recently, the hybridization of PSO and EDAs is emerged as a new research trend. A mentionable proposal is EDA-PSO [5] where the next generation population is generated using PSO or EDA method basing on a continuously updated probability. Instead, in [6] the population is splitted into chunks, hence the intra-chunk update follows the PSO rules, while the global update follows EDAs schema. Another approach is EDPSO [7], which also relies on an Ant Colony Optimization (ACO) variant for continuous domains [8]. Finally, we mention PSO_bounds [9], which uses EDAs schema to manipulate the allowable bounds for the PSO particles.

Conversely from the various PSO-EDA hybridization proposals, our algorithm (PSEDA) can be seen as a PSO implementation in the EDAs framework. The genotype of a PSEDA individual shares various properties with that of a PSO particle. Furthermore, PSEDA and PSO base their behavioural dynamics on the same attractive points. The notable difference with respect to PSO is the absence of velocities, while the main innovation with respect to usual EDAs is the learning of an independent probability distribution model for each individual.

The paper is organized as follows: a brief background about PSO and EDAs are covered in Sections 2 and 3. The proposed PSEDA is explained in Section 4, while some experimental results about it are provided in Section 5. Finally, the paper is concluded in Section 6 where, other than some considerations, future work guidelines for some possible industrial applications are reported.

2 Particle Swarm Optimization

PSO has been introduced by Kennedy and Eberhart [1] and got its inspiration from particles model of objects and simulation of the collective behaviour of flocks of birds.

A PSO swarm is composed by a set of n particles $P = \{p_1, p_2, \ldots, p_n\}$ interconnected in a graph that defines a neighborhood relation among them, i.e. for each particle p_i its neighborhood set $N_i \subseteq P$ is defined (in the following we assume, as it is in general, that $p_i \in N_i$). The position of a particle represents a candidate solution of the given optimization problem represented by the objective/fitness function $f : \Theta \to \mathbb{R}$ with $\Theta \subseteq \mathbb{R}^d$ (the region of feasible solutions) to be minimized (or maximized). Each particle, as time passes, through its search adjusts its position according to its own experience as well as the experience of its neighbours. In this way PSO combines cognitive and social strategies in order to focus the search of the swarm toward the most promising areas.

At any time step t (with $t \in \mathbb{N}$) the *genotype* of a particle p_i is formed by the following d-dimensional vectors (where d is the dimensionality of the search space):

- $x_{i,t}$, i.e. the particle position,
- $v_{i,t}$, i.e. the particle velocity,
- $b_{i,t}$, i.e. the particle personal best position ever visited until time step t,

- $l_{i,t}$, i.e. the best position ever found among its neighbours until time step t.

In the more usual PSO implementation, a complete network is adopted as neighborhood graph. In this case the $l_{i,t}$ values are replaced by a global g_t that is the same for all the particles thus improving the efficiency of the algorithm. We will call this implementation *global PSO* (gPSO).

PSO starts by assigning random positions within the feasible region Θ. Furthermore, velocities are usually initialized to small random values in order to prevent particles from leaving the search space in the early iterations.

During a main loop, velocities and positions are iteratively updated until a stop criterion is met (e.g. a given amount of fitness evaluations has been performed). The update rules are:

$$x_{i,t+1} = x_{i,t} + v_{i,t+1} \tag{1}$$

$$v_{i,t+1} = \omega v_{i,t} + \varphi_1 r_{1,t}(b_{i,t} - x_{i,t}) + \varphi_2 r_{2,t}(l_{i,t} - x_{i,t}) \tag{2}$$

Weights in (2) respectively represent the inertia ω and the acceleration factors φ_1 and φ_2. Instead, $r_{1,t}$ and $r_{2,t}$ are two random numbers uniformly distributed in $[0,1]$. The three terms of the velocity update rule (2) characterize the behaviour of the particles. The first term, called *inertia* or momentum, serves as a memory of the previous flight trajectory and prevents a particle from drastically changing direction. The second term is the *cognitive component* that models the tendency of the particles to return to the previously found personal best position. Finally, the third term represents the *social component* and quantifies the velocity contribution of the neighbours (or of the entire swarm in the gPSO case).

The (personal and social) best positions are updated whenever there is an improvement in fitness. More formally (in the case of a minimization problem):

$$b_{i,t+1} = \begin{cases} x_{i,t+1} & \text{if } f(x_{i,t+1}) \leq f(b_{i,t}) \\ b_{i,t} & \text{otherwise} \end{cases} \tag{3}$$

$$l_{i,t+1} = \arg \min_{j:p_j \in N_i} f(b_{j,t+1}) \tag{4}$$

In the gPSO case the assignment (4) becomes:

$$g_{t+1} = \arg \min_{1 \leq i \leq n} f(b_{i,t+1}) \tag{5}$$

Sometimes the particles can exceed the bounds of the feasible search space Θ. There are several solutions to this problem [10], but generally the preferred one is to randomly reset an out-of-bounds component between the previous position and the bound exceeded according to:

$$x_{i,t+1,k} = \begin{cases} x_{i,t,k} + r(u_k - x_{i,t,k}) & \text{if } x_{i,t+1,k} > u_k \\ x_{i,t,k} - r(x_{i,t,k} - l_k) & \text{if } x_{i,t+1,k} < l_k \end{cases} \tag{6}$$

where r is a random number in $[0,1]$, k is the index of the exceeding component and l_k, u_k are, respectively, the lower and upper bounds for this component. Conversely

of a completely random restart, in this way the original trajectory of the particle is somehow preserved.

Finally, note that a lot of studies on the PSO parameters tuning have been done. However, in [11] it has been proved that a good general setting, able to allow a convergence without the need of bounds for the velocity components, is the following: $\omega = 0.7298$, $\varphi_1 = \varphi_2 = 1.49618$.

3 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) [3] are a recently new schema of evolutionary algorithms (EAs). The main difference with the other EAs is that EDAs rely on a probabilistic model and do not use variation operators such as crossover or mutation.

They select some solutions from the current population and perform a learning procedure on them. In this learning phase EDAs build a *probability distribution model* (PDM) basing on some selected solutions. Successively, the solutions for the next generation are sampled from that model.

More precisely, EDAs perform the following steps:

0. Randomly generate n initial solutions.
1. Select $m \leq n$ solutions from the current population according to a selection method.
2. Build a probability model PDM basing on the m selected solutions.
3. Replace $q \leq n$ members of the current population by new solutions sampled from the probability model PDM.
4. If a stop criterion is not met (e.g. a given amount of fitness evaluations has been performed), go to step 1.

The selection method in step 1 usually chooses the best m solutions basing on their fitness, although sometimes a stochastic method (like a fitness proportional selection) is preferred in order to maintain a certain degree of diversity. Instead, the replacement method in step 3 generally substitutes the q worst members, so the EDA presents an elitist behaviour[1].

The model built in step 2 generally assumes the form of a set of probability distributions, one for each dimension of the search space. For example, in the case that the genotype is a bitstring of length d, the PDM could consist in a vector of d probability values p_i (with $1 \leq i \leq d$) where each p_i is the probability that the ith bit is a 1. Sometimes the PDM is a multivariate probability distribution, but, for our purposes, this is not the case.

Finally, we report that, although EDAs have been originally introduced to tackle combinatorial optimization problems, recently, numerical applications have also been proposed [4].

[1] Note that there is no elitism if $q = n$.

4 Particle Swarm Estimation of Distribution Algorithm

Particle Swarm Estimation of Distribution Algorithm (PSEDA), more than an hybridization between PSO and EDAs, can be seen as a PSO implementation in the EDAs framework. The notable difference with respect to PSO is the absence of velocities, while the main innovation with respect to usual EDAs is the learning of an independent probability distribution model (PDM) for each individual (other than for each dimension).

In order to understand how the PSO position update process is simulated in PSEDA, we first analyze the PSO rules (1) and (2). In (1) the new particle position depends on its current position and on its velocity, which in turn is a function of the previous velocity, the personal best position, the social best position, and, again, the current position (see rule (2)). Summarizing and simplifying, we can say that $x_{i,t+1}$ depends on $x_{i,t}$, $b_{i,t}$, and $l_{i,t}$. Note that this is a coarse simplification since we have omitted the previous velocity value $v_{i,t}$, however, this behaviour is somehow reintroduced later in the probability model.

Before explaining the PSEDA learning procedure, we note that, as usual in continuous optimization, the objective function f is provided with its feasible region $\Theta = [l_k, u_k]^d \subseteq \mathbb{R}^d$ (with $1 \leq k \leq d$).

As aforementioned, in PSEDA, a PDM is independently built for each one of the n population individuals. Let d be the dimensionality of the search space, then each PDM is modeled by d mixtures of (univariate) probability distributions (i.e. a mixture for each dimension of the problem). The probability distributions composing the mixture $M_{i,t,k}$ of the kth dimension of individual i at time step t are:

1. $TN^x_{i,t,k}(x_{i,t,k}, \sigma_t; l_k, u_k)$, i.e. a Gaussian distribution, with mean $x_{i,t,k}$ and standard deviation σ_t, truncated in $[l_k, u_k]$,
2. $TN^b_{i,t,k}(b_{i,t,k}, \sigma_t; l_k, u_k)$, i.e. a Gaussian distribution, with mean $b_{i,t,k}$ and standard deviation σ_t, truncated in $[l_k, u_k]$,
3. $TN^l_{i,t,k}(l_{i,t,k}, \sigma_t; l_k, u_k)$, i.e. a Gaussian distribution, with mean $l_{i,t,k}$ and standard deviation σ_t, truncated in $[l_k, u_k]$,
4. $U_k(l_k, u_k)$, i.e. a continuous uniform distribution in $[l_k, u_k]$,
5. $\overline{M}_{i,t-1,k}(x_{i,t-1,k}, b_{i,t-1,k}, l_{i,t-1,k}; \sigma_t; l_k, u_k)$, i.e. a reduced version of the previous mixture distribution for the same dimension of the same particle. This mixture is composed by the three truncated (in $[l_k, u_k]$) Gaussians with means $x_{i,t-1,k}$, $b_{i,t-1,k}$, $l_{i,t-1,k}$ and standard deviation σ_t.

At the end of every generation, once built the PDMs, a new individual position is sampled from the mixture distributions that form its PDM.

In this way, the three truncated Gaussians (TN) ensure that the attractive points of PSO ($x_{i,t}$, $b_{i,t}$, $l_{i,t}$) are attractive also for PSEDA (since they are the means of the Gaussians). The uniform component of the mixture (U) is introduced in order to regulate the exploration degree of PSEDA. Finally, the last component of the mixture $\overline{M}_{i,t-1,k}$, a relaxed version of the previous mixture $M_{i,t-1,k}$, tries to play the role of the previous velocity in PSO rule (2).

In each one of the (truncated) Gaussian distributions, at time step t, a common standard deviation σ_t is used. This parameter, although in a different way from the uniform distribution U_k, is significant for the exploration/exploitation balance of the algorithm. In order to match the more common EAs best practices [12], σ_t should be relatively large in early generations (favoring exploration) and relatively small in later generations (favoring exploitation). For this reason we have decided to shade σ_t from a relatively great to a relatively small value with the passing of generations. After some preliminary experiments we have chosen to shade σ_t from $1/10$ to $1/10^6$ of the dimension length $(u_k - l_k)$. More formally:

$$\sigma_t = \frac{u_k - l_k}{t\frac{10^6-10}{G} + 10} \tag{7}$$

where G is the maximum number of generations allowed in a computation.

As well known, the probability density function (*pdf*) of a mixture distribution is a convex combination of the *pdf*s of its component distributions [13]. Let $g_x(z)$, $g_b(z)$, $g_l(z)$, $g_u(z)$, $g_m(z)$ be the *pdf*s of the five component distributions (note that, in order to have a clear notation, we have omitted some indices), then the *pdf* $m(z)$ of the mixture distribution $M_{i,t,k}$ is:

$$m(z) = w_x g_x(z) + w_b g_b(z) + w_l g_l(z) + w_u g_u(z) + w_m g_m(z) \tag{8}$$

where the weights w_x, w_b, w_l, w_u, w_m are non-negative values that sum up to 1[2]. Hence we have five weights against the three of PSO. However, as illustrated in Section 5, we can set them proportionally with respect to the PSO weights (i.e.: ω, φ_1, φ_2) that we would have chosen for the given problem. In this way, the PSO concepts of inertia, cognitive acceleration and social acceleration are even more preserved.

Summarizing, the genotype of a PSEDA individual is composed, other than by the current position $x_{i,t}$, the personal best $b_{i,t}$, and the social best $l_{i,t}$, also by the same values at the previous generation, i.e. $x_{i,t-1}$, $b_{i,t-1}$, $l_{i,t-1}$. Indeed, these are needed for computing the distribution $\overline{M}_{i,t-1,k}$[3]. The complete algorithm for PSEDA is shown in Algorithm 1.

In PSEDA, exactly as in PSO, in order to have the social best values, a neighborhood relation among the individuals must be defined. In the case that a complete network is adopted as neighborhood graph, the $l_{i,t}$ values can be replaced by a global g_t thus improving the space/time efficiency of the algorithm. Similarly as in PSO, we call this implementation gPSEDA.

Finally, note that a mixture distribution can be easily sampled as explained in [13], while a truncated Gaussian can be sampled either using the accept-reject method (more suitable when σ_t is small) or the inverse cumulative function method (more suitable when σ_t is large) [14]. With this approach, out-of-bounds individuals

[2] Note that, for sampling $\overline{M}_{i,t,k}$, only the weights w_x, w_b, w_l are needed. Hence, in this case, they are temporarily normalized in order to sum up to 1.

[3] This genotype richness ensures an elitist behaviour at the algorithm.

Algorithm 1 PSEDA

```
1: procedure PSEDA
2:     Randomly generate n initial solutions and evaluate them
3:     t ← 1
4:     while t ≤ max_t do
5:         for all individuals i do
6:             for all dimensions k do
7:                 Learn the mixture distribution M_{i,t,k}
8:             end for
9:             Store previous positions x_{i,t−1}, b_{i,t−1}, l_{i,t−1}
10:            for all dimensions k do
11:                Sample x_{i,t,k} from M_{i,t,k}
12:            end for
13:            Evaluate individual i
14:            Update personal best b_{i,t}
15:        end for
16:        Update social best l_{i,t}s
17:        Update standard deviation σ_t
18:        t ← t + 1
19:    end while
20: end procedure
```

become no more possible, hence, repair procedures, as those used in PSO, are not needed.

5 Experiments

The performances of PSEDA have been evaluated on a suite of eight well known benchmark functions. These functions are non-noisy, non-translated, non-rotated versions of those proposed in [15] for the real-parameters optimization competition at CEC 2005. They are: sphere, ellipsoid, Rosenbrock, Griewank, Ackley, Rastrigin, Weierstrass, and extended Schaffer. This suite presents a variegate combination for what regards the properties of modality and separability of the functions composing it.

Each benchmark is investigated with dimensionality $d = 10$. An execution is regarded convergent if $f(x) - f(x^{opt}) < \varepsilon$, on the other hand, the execution will be aborted if the number of function evaluations (NFES) exceeds the allowed cap of $300,000$. In Table 1, for each benchmark, its function definition, the optimization interval and the ε value used for it are reported.

The performances of a fully connected PSEDA (gPSEDA) are compared with those of a classical fully connected PSO (gPSO). The gPSO parameters are those widely used in literature and suggested in [11], i.e.: $\omega = 0.7298$, $\varphi_1 = \varphi_2 = 1.49618$. Also gPSEDA has some parameters to be set. After some preliminary tests, and inspired from the usual values for the mutation rate parameter of classical genetic algorithms [16], we have set the exploration weight w_u to a small value, i.e. $w_u =$

Table 1 Experiments Setup

Function Definition	Opt. Interval	ε
$f_1(x) = \sum_{i=1}^{d} x_i^2$	$[-100,100]^d$	10^{-6}
$f_2(x) = \sum_{i=1}^{d} (10^6)^{\frac{i-1}{d-1}} x_i^2$	$[-100,100]^d$	0.5
$f_3(x) = \sum_{i=1}^{d-1} \left(100(x_i^2 - x_i + 1)^2 + (x_i - 1)^2\right)$	$[-100,100]^d$	10^{-6}
$f_4(x) = \sum_{i=1}^{d} x_i^2/4000 - \prod_{i=1}^{d} \cos(x_i/\sqrt{i}) + 1$	$[-300,300]^d$	0.5
$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	$[-100,100]^d$	10^{-2}
$f_6(x) = \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5,5]^d$	10^{-2}
$f_7(x) = \sum_{i=1}^{d} \sum_{k=0}^{20} 0.5^k \cos(2\pi \cdot 3^k \cdot (x_i + 0.5)) - d\sum_{k=0}^{20} 0.5^k \cos(2\pi \cdot 3^k \cdot 0.5)$	$[-0.5,0.5]^d$	0.5
$f_8(x) = \sum_{i=1}^{n-1} S(x_i, x_{i+1}) + S(x_n, x_1)$, where $S(x,y) = 0.5 + \frac{\sin^2(\sqrt{x^2+y^2}) - 0.5}{(1+0.001(x^2+y^2))^2}$	$[-100,100]^d$	0.5

0.05. Instead, since the other parameters have a similar meaning of those of PSO, we have decided to set them proportionally with respect to those used for gPSO. Let $s = \omega + \varphi_1 + \varphi_2$, then $w_x = (1 - w_u) \cdot 0.5 \cdot \frac{\omega}{s} = 0.09313$, $w_b = (1 - w_u) \cdot \frac{\varphi_1}{s} = 0.38187$, $w_l = (1 - w_u) \cdot \frac{\varphi_2}{s} = 0.38187$, $w_m = (1 - w_u) \cdot 0.5 \cdot \frac{\omega}{s} = 0.09313$. Other than consistent with the usual PSO parameters setting, our choice is resulted better than other settings that we have tried in some preliminary experiments (however, more systematic experiments are still needed).

In order to eliminate the randomness of the statistical results, for each benchmark 30 executions have been held. In each experiment, the convergence probability P_c (i.e. the number of convergent executions above the total number of executions) and the average NFES of all convergent executions C are recorded. These two indices are also synthesized in the quality measure $Q_m = C/P_c$ suggested in [15].

Table 2 Experimental Results

Benchmark	gPSEDA			gPSO		
	C	P_c	Q_m	C	P_c	Q_m
f_1	66788	1.00	66788	4227	1.00	4227
f_2	71651	1.00	71651	5631	0.93	6035
f_3	-	0.00	-	214373	0.10	2143730
f_4	1787	1.00	1787	2507	0.97	2585
f_5	5447	1.00	5447	2527	1.00	2527
f_6	8397	1.00	8397	-	0.00	-
f_7	8263	1.00	8263	3050	0.20	15250
f_8	6234	0.97	6447	-	0.00	-

In Table 2, the performance indices C, P_c, and Q_m are reported both for gPSEDA and for gPSO. The results show that gPSEDA is not as efficient as gPSO on the unimodal benchmarks (f_1, f_2) and on the Rosenbrock function (f_3), a multimodal function with a very small number of optima. However, on the other multimodal benchmarks, gPSEDA clearly outperforms gPSO both in convergence (C) than in probability of convergence (P_c) (apart on Ackley function f_5, an historically good benchmark for PSO, where the results are anyhow similar). The best results are obtained on Rastrigin function f_6 and on extended Schaffer function f_8, where our approach is able to reach a near optimum solution with a small amount of NFES and with a full convergence. Instead gPSO, on these benchmarks was never able to reach a near optimum solution before the allowed cap of NFES. Since f_6 is a separable function and f_8 is a non-separable function, we can conclude that PSEDA, with respect to PSO, is very versatile, other than efficient, on these problems that present a certain complexity on their landscapes.

6 Conclusion and Future Work

In this paper Particle Swarm Estimation of Distribution Algorithm (PSEDA), an hybridization of PSO and EDAs, has been introduced by implementing the PSO dynamics in the EDAs framework. The velocity concept of PSO has been simulated using the PSO attractive positions as topical points in a probability distribution model.

Regarding PSEDA from the viewpoint of EDAs, the innovation introduced is the learning (and hence the sampling) of an independent probability distribution model for each individual (other than for each dimension). Furthermore, conversely from classical EDAs, despite the entire population update from one generation to the next, the individual genotype richness allows anyway an elitist behaviour to PSEDA.

Experimental results show that PSEDA improves the performances with respect to classical PSO on problems that present a certain complexity on their landscapes. Further experiments are needed to confirm the positive results on a wider range of benchmarks, such as rotated, translated and noisy functions. Furthermore, analysis of the parameters setting and of the standard deviation regulation are planned in order to provide a better understanding of the PSEDA behaviour.

A good future work guideline is also the discretization of PSEDA. Indeed, since it uses an independent probability distribution model for each population individual, PSEDA can be easily converted to handle discrete search spaces by using mixtures of discrete probability distributions. For example, the continuous Gaussian distributions can be replaced by discrete binomial distributions in the case of ordered discrete search spaces, or by probability histograms of the form ”needle in a haystack” in the case of purely combinatorial search space (those without an ordered structure). By combining these two approaches, also partially ordered domains can be handled. This discretization process, once formalized and experimented, jointly with the continuous PSEDA described in this paper, could allow to handle several prac-

tical applications, that generally consist of hybrid continuous/combinatorial spaces (i.e. search spaces where some dimensions are continuous and other ones are combinatorial). Imagine, for example, the automatic testing scenario where an industry, before commercializing an engineering product, must test the product in order to optimize some features (e.g. the aerodynamics of an airplane that is repetitively tested with a scale model in an automatized and robotized wind tunnel using a sort of trial-and-error method). Finally, another practical application, already approached with a similar technique [17], is the optimization of web content presentations using the users feedback as an online fitness function to optimize.

References

1. J. Kennedy, R. Eberhart, "Particle swarm optimization", in Proc. of IEEE Conf. on Neural Networks, IEEE Press, 1995, pp. 1942-1948
2. B. Jarboui, M. Cheikh, P. Siarry, A. Rebai, "Combinatorial particle swarm optimization (CPSO) for partitional clustering problem", Applied Mathematics and Computation, Vol. 192, Issue 2, pp. 337-345
3. Larranaga, P. and Lozano, J. A., "Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation", Kluwer Academic Publishers, 2001
4. Bosman, P. A. N. and Thierens, D., "Expanding from Discrete to Continuous EDAs: The IEDA", Proceedings of Parallel Problem Solving from Nature, PPSN-VI, pp. 767-776, 2000
5. Y. Zhou and J. Jin, "Eda-pso - a new hybrid intelligent optimization algorithm", in Proc. of the Michigan University Graduate Student Symposium, 2006
6. Bengoetxea, E., Larranaga, P., "EDA-PSO: A Hybrid Paradigm Combining Estimation of Distribution Algorithms and Particle Swarm Optimization", Swarm Intelligence, Springer LNCS, pp. 416-423, 2010
7. M. Iqbal and M. A. Montes de Oca, "An estimation of distribution particle swarm optimization algorithm", in Proc. of the Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence, 2006, pp. 7283
8. K. Socha and M. Dorigo, "Ant colony optimization for continuous domains", Universite Libre de Bruxelles, IRIDI Technical Report Series, Technical Report No. TR/IRIDIA/2006-037
9. El-Abd, Mohammed, Kamel, Mohamed, "PSO_Bounds: A New Hybridization Technique of PSO and EDAs", Foundations of Computational Intelligence Vol. 3, pp. 509-526, 2009
10. Price, K V, R M Storn, e J A Lampinen, "Differential evolution: a practical approach to global optimization", Springer Verlag, 2005
11. Clerc, M., Kennedy, J., "The particle swarm-explosion, stability, and convergence in a multidimensional complex space", IEEE Transaction on Evolutionary Computation, 6(1), pp. 58-73, 2002
12. T. Bäck, D. Fogel, and Z. Michalewicz, "Handbook of evolutionary computation", Oxford Univ. Press, 1997
13. Titterington, D., A. Smith, and U. Makov "Statistical Analysis of Finite Mixture Distributions," John Wiley & Sons, 1985
14. Luc Devroye, "Non-Uniform Random Variate Generation", Springer Verlag, 1986
15. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization", Technical Report, Nanyang Technological University, Singapore, May 2005
16. Michalewicz, Z., "Genetic algorithms + data structures = evolution programs", Springer, 1992
17. Santucci V., Milani A., Leung C., "Optimal Design of Web Information Contents for E-Commerce Applications", Proceedings of the 25th International Symposium on Computer and Information Sciences, Springer LNEE, Vol. 62, pp. 339-344