PSO_Bounds: A New Hybridization Technique of PSO and EDAs

Mohammed El-Abd and Mohamed S. Kamel

Abstract. Particle Swarm Optimization (PSO) is a nature inspired population-based approach successfully used as an optimization tool in many application. Estimation of distribution algorithms (EDAs), are evolutionary algorithms that try to estimate the probability distribution of the good individuals in the population. In this work, we present a new PSO algorithm that borrows ideas from EDAs. This algorithm is implemented and compared to previous PSO and EDAs hybridization approaches using a suite of well-known benchmark optimization functions.

1 Introduction

Particle Swarm Optimization (PSO) [1, 2] is an optimization method widely used to solve continuous nonlinear functions. Although, the original intent was to simulate the movement of a flock of birds or a school of fish looking for food, It was soon realized that the associated equations of motion could be used as a very powerful optimization tool.

Estimation of distribution algorithms (EDAs) [3] are evolutionary algorithms that solve the problem in hand by trying to build a probabilistic model that estimates the distribution of good regions in the search space. These algorithms work by continuously updating the generated model and using it to produce new solutions. One of the early works in this are is Population-Based Incremental Learning (PBIL) proposed in [4]. PBIL is an optimization method similar to Genetic algorithms but with maintaining a probabilistic model rather than a population of solutions. This model was updated in every generation and was used to produce the next population.

Mohammed El-Abd and Mohamed S. Kamel
ECE Dept., University of Waterloo, 200 University Av. W., Waterloo, Ontario, Canada, N2L3G1

A. Abraham et al. (Eds.): Foundations of Comput. Intel. Vol. 3, SCI 203, pp. 509–526.
springerlink.com

In the past few years, two hybrid models that mix the PSO algorithm with EDAs have been proposed in the literature [5, 6]. Both approaches use the same probabilistic model to describe the search space but differ in the way the information is gathered and used to build the model. They also differ in the way the model is used to generate new solutions.

In this work, we propose a new model that is based on PBIL. This model continuously use the distribution of the particles during the search to update the bounds of the PSO search space. This in turn affects the particles movement as it affects both the bounds of allowable movement and the maximum allowable velocity. The proposed algorithm is compared to other hybrid techniques and is shown to outperform them on the more difficult multimodal functions.

The chapter is organized as follows: a brief background on PSO is given in Section 1.2. This is followed by an introduction to EDAs in Section 1.3. A literature review of previous PSO and EDAs hybridization techniques are covered in Section 1.4. The new algorithm is proposed in Section 1.5. Results and discussions are presented in Section 1.6. The chapter is concluded in Section 1.7.

2 Particle Swarm Optimization

PSO [1, 2] is regarded as a population-based method, where the population is referred to as a swarm. The swarm consists of a number of individuals called particles. Each particle i in the swarm holds the following information:

- The current position x_i,
- The current velocity v_i,
- The best position, the one associated with the best fitness value the particle has achieved so far $pbest_i$,
- The global best position, the one associated with the best fitness value found among all of the particles $gbest$.

In every iteration, each particle adjusts its own trajectory in the space in order to move towards its best position and the global best according to the following equations:

$$v_{ij}^{t+1} = wv_{ij}^{t} + c_1 r_{1j}^{t}(pbest_{ij}^{t} - x_{ij}^{t}) + c_2 r_{2j}^{t}(gbest_{j}^{t} - x_{ij}^{t}), \quad (1)$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1}, \quad (2)$$

for $j \in 1..d$ where d is the number of dimensions, $i \in 1..n$ where n is the number of particles, t is the iteration number, w is the inertia weight, r_1 and r_2 are two random numbers uniformly distributed in the range [0,1], and c_1 and c_2 are the acceleration factors.

Afterwards, each particle updates its personal best using the equation (assuming a minimization problem):

$$pbest_i^{t+1} = \begin{cases} pbest_i^t & \texttt{if} \quad f(pbest_i^t) \leq f(x_i^{t+1}) \\ x_i^{t+1} & \texttt{if} \quad f(pbest_i^t) > f(x_i^{t+1}) \end{cases} \tag{3}$$

Finally, the global best of the swarm is updated using the equation (assuming a minimization problem):

$$gbest^{t+1} = \arg \min_{pbest_i^{t+1}} f(pbest_i^{t+1}), \tag{4}$$

where *f(.)* is a function that evaluates the fitness value for a given position. This model is referred to as the *gbest* (global best) model.

Another model is the *lbest* (local best) model [7], in each particle does not hold the global best position. Instead, each particle only holds the best position achieved by its own neighborhood. Different neighborhood structures were previously examined for such a model [8] including the ring topology and the Von Neumann model.

3 Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDAs) are evolutionary algorithms that try to estimate the probability distribution of the good individuals in the population. EDAs try to estimate this probability distribution by using selected individuals, from the current population, to construct a probabilistic model. This model is consequently used to generate the offspring. The new population is generated by selecting individuals from both the offspring and the current population in a proportionate manner. Finally, the new population replaces the current one. Hence, EDAs maintain a continuously updated probabilistic model from one generation to the next. Although, it has been originally introduced to tackle combinatorial optimization problems, recent numerical applications have been proposed as well [9, 10, 11, 12]. The general steps for an EDA is shown in Algorithm 1.

EDAs are categorized based on the degree of dependencies, allowed by the probabilistic model used, between the problem variables:

- No dependency: the problem variables are assumed to be independent,
- Bivariate dependency: the dependencies are only assumed between two variables at a time,
- Multivariate dependency: the dependencies could be modeled between any number of variables.

For a complete survey of the different optimization techniques adopted using building probabilistic models, the interested reader could refer to [13].

Algorithm 1. Estimation of Distribution Algorithm (EDA)

```
1: P ⇐ Initialize the population
2: Evaluate the initial population
3: while iter_number ≤ Max_iterations do
4:    P_s ⇐ Select the top s individuals
5:    M ⇐ Estimate a new Model from P_s
6:    P_n ⇐ Sample n individuals from M
7:    Evaluate P_n
8:    P ⇐ Select n individuals from PUP_n
9:    iter_number = iter_number + 1
10: end while
11: return Best_Individual
```

4 PSO Based on Probabilistic Models

This section surveys the two previous attempts to introduce the concepts of EDAs into PSO in order to improve its performance.

4.1 EDPSO

An estimation of distribution particle swarm optimizer (EDPSO) was proposed by Iqbal and Montes de Oca [5]. The method borrowed some ideas from a development in ACO for solving continuous optimization problems [14, 15]. The approach relies on estimating the joint probability distribution for one dimension at a time using mixtures of weighted Gaussian functions. The Gaussian functions are defined through an archive of k solutions ($pbests$ of the particles). For each dimension d, the dimension is either updated using PSO equations or by sampling a Gaussian distribution selected from the archive. The values of this dimension d across all the solutions in the archive compose the vector $\boldsymbol{\mu_d}$, which is the vector of means for the univariate Gaussian distributions:

$$\boldsymbol{\mu_d} =< pbest_{1d}, pbest_{2d}, ..., pbest_{kd} > \tag{5}$$

To select one of these distributions, the weights vector $\mathbf{w}$, which holds the weights associated with each distribution, is calculated. This is done by sorting the solutions according to their fitness, with the best solution having a rank of 1. A weight is calculated for each solution as follows:

$$\mathbf{w} =< w_1, w_2, ..., w_k > \tag{6}$$

$$w_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{7}$$

which is a Gaussian function with mean l and standard deviation qk, where q is a constant that determines how much we prefer good solutions and l is the solution rank.

The Gaussian function to be used is selected probabilistically. The probability of selecting a certain Gaussian function is proportional to its weight. This probability is calculated as follows:

$$\mathbf{p} =< p_1, p_2, ..., p_k >$$
$$p_l = \frac{w_l}{\sum_{r=1}^{k} w_r} \tag{8}$$

After selecting a certain Gaussian function G_d denoted by its mean $pbest_{gd}$, where $1 \leq g \leq k$, the standard deviation for this functions is calculated as:

$$\sigma_{gd} = \xi \sum_{i=1}^{k} \frac{|pbest_{id} - pbest_{gd}|}{k-1} \tag{9}$$

which the average distance between the selected mean and the other entries of the archive. ξ is a parameter to balance the exploration-exploitation behaviors. if ξ is small, this will lead to having a smaller value for σ_{gd} and the search will tend to search in a closer range around the chosen mean.

Finally the selected Gaussian function is evaluated (not sampled) to generate a value r in order to probabilistically move the particle. This is done by generating a uniformly distributed random number U(0,1). If it is less than r, the particle moves using the normal PSO equations. Otherwise, the Gaussian function is sampled to move the particle. The steps are shown in Algorithm 2.

4.2 EDA-PSO

A hybrid EDA-PSO approach was proposed in [6]. The algorithm works by sampling an independent univariate Gaussian distribution based on the best half of the swarm. The mean and standard deviation of the model is calculated in every iteration as:

$$\boldsymbol{\mu} = \frac{1}{M} \sum_{i=1}^{M} \mathbf{x}_i \tag{10}$$

$$\sigma_j = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (\mathbf{x}_{ij} - \mu_j)^2}, \tag{11}$$

where $M = N/2$ for a swarm with N particles and i is the particle number.

The choice of whether to update the particle using the normal PSO equations or to sample the particle using the estimated distribution is made with a probability p, referred to as the *participation ratio*. If $p = 0$, the algorithm will behave as a pure EDA algorithm. On the other hand, if $p = 1$, it will be a pure PSO algorithm. In the hybrid approach, where $0 < p < 1$, each

Algorithm 2. The EDPSO algorithm.

Require: Max_Function_Evaluations
Initialize the swarm
Max_Iterations = $\frac{Max\_Function\_Evaluations}{Num\_Particles}$
iter_number = 1
while $iter\_number \leq Max\_Iterations$ **do**
  Update the swarm
  Rank the particle's using pbests information
  Calculate weights vector **w**
  Calculate probabilities vector **p**
  for every particle i **do**
    for each dimension d **do**
      Update v_{id} and x_{id}
      Select a Gaussian function according to p_i
      Calculate σ_{gd}
      Prob_move = $\sigma_{gd}\sqrt{2\pi}G_d(x_{id})$
      if $U(0,1) < Prob\_move$ **then**
        continue
      else
        x_{id} = Gauss(s_{gd},σ_{gd})
      end if
    end for
  end for
  iter_number = iter_number + 1
end while
return *gbest*

particle is either totally updated by the PSO equations or totally sampled from the estimated distribution (not on a dimension-by-dimension basis as in EDPSO). Finally, the particle gets updated only if its fitness improves.

The authors also proposed different approaches in order to adaptively set the parameter p. These approaches depend on the success rate of both the PSO and EDA parts in improving the particles' fitness:

- The *Generation based*, where the success rates are calculated based on the information gathered during the last generation,

$$p^{t+1} = \frac{\frac{sum\_PSO^t}{num\_PSO^t}}{\frac{sum\_PSO^t}{num\_PSO^t} + \frac{sum\_EDA^t}{num\_EDA^t}} \tag{12}$$

- The *All historical information*, where the success rates are calculated based on the information gathered during the entire search,

$$p^{t+1} = \frac{\sum_{i=1}^{t} \frac{sum\_PSO^i}{num\_PSO^i}}{\sum_{i=1}^{t} \frac{sum\_PSO^i}{num\_PSO^i} + \sum_{i=1}^{t} \frac{sum\_EDA^i}{num\_EDA^i}} \tag{13}$$

Algorithm 3. The EDA-PSO algorithm

Require: Max_Function_Evaluations
Initialize the swarm
Max_Iterations = $\frac{Max\_Function\_Evaluations}{Num\_Particles}$
iter_number = 1
while $iter\_number \leq Max\_Iterations$ **do**
  Calculate $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ using top $\frac{N}{2}$ particles
  for every particle i **do**
    if $U(0,1) < p$ **then**
      candidate_particle = PSO equations
    else
      candidate_particle = Gauss($\boldsymbol{\mu}$,$\boldsymbol{\sigma}$)
    end if
    if candidate_particle has a better fitness **then**
      particle i = candidate_particle
    end if
  end for
  iter_number = iter_number + 1
end while
return *gbest*

- The *Sliding window*, where the success rates are calculated considering only the information in the last m generations.

$$p^{t+1} = \frac{\sum_{i=t-m+1}^{t} \frac{sum\_PSO^i}{num\_PSO^i}}{\sum_{i=t-m+1}^{t} \frac{sum\_PSO^i}{num\_PSO^i} + \sum_{i=t-m+1}^{t} \frac{sum\_EDA^i}{num\_EDA^i}} \tag{14}$$

In all the previous equations sum_PSO^t and num_PSO^t refers to the sum of improvements and number of improvements done by the PSO component at iteration t. While sum_EDA^t and num_EDA^t refers to the sum of improvements and number of improvements done by the EDA component at iteration t. Finally, m is the window size.

The complete algorithm for EDA-PSO is shown in Algorithm 3.

5 PSO with Varying Bounds

A PBIL approach for continuous search spaces was proposed in [10]. The algorithm explored the search space by dividing the domain of each gene into two equal intervals referred to as the *low* and *high* intervals. A probability h_d, which is initially set to 0.5, is the probability of gene number d being in the *high* interval as shown:

$$x_d \in [a,b], h_d = Probability(x_d > \frac{a+b}{2}) \tag{15}$$

After each generation, this distribution is updated according to the gene values of the best individual using the following formula:

$$p = \begin{cases} 0 & \texttt{if} \quad x_d^{max} < \frac{a+b}{2} \\ 1 & otherwise \end{cases} \tag{16}$$

$$h_d^{t+1} = (1-\alpha) * h_d^t + \alpha * p \tag{17}$$

where α is the *relaxation factor* and t is the iteration number. If h_d gets below h_{dmin} or above h_{dmax}, the population gets re-sampled in the corresponding interval, $[a, \frac{a+b}{2}]$ or $[\frac{a+b}{2}, b]$, respectively.

In this work, we propose a new PSO algorithm, referred to as PSO_Bounds, which borrows concepts from PBIL. At the beginning, the particles are initialized in the predefined domain. After every iteration, the probability h_d of each dimension d gets adjusted according to the probability of this dimension value being in the *high* interval of the defined domain. This probability is calculated using information from all the particles and not only *gbest* to prevent premature convergence. Hence, the original equations of PBIL are changed as follows:

$$p_{id}^t = \begin{cases} 0 & \texttt{if} \quad pbest_{id}^t < \frac{a+b}{2} \\ 1 & otherwise \end{cases} \tag{18}$$

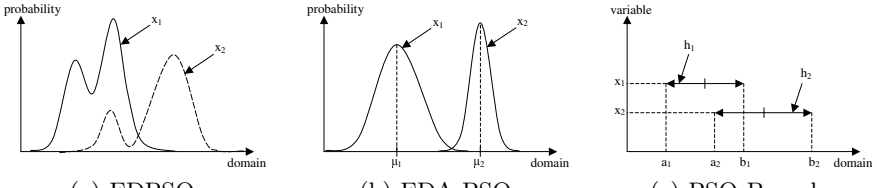
$$p_d^t = \frac{\sum_i^n p_{id}^t}{n} \tag{19}$$

$$h_d^{t+1} = (1-\alpha) * h_d^t + \alpha * p_d^t \tag{20}$$

where $i \in 1..n$ where n is the number of particles, t is the iteration number, and d is the dimension. Please note that these equations are applied for each dimension d separately.

In PBIL, the probabilities were updated using the value of the best individual, which is analogous to the current position of the particles in PSO. However, in our implementation, we use the values of *pbests* instead as it reflects the best experience of the swarm and would guide the search towards better solutions.

When h_d^{t+1} gets specific enough, the domain of dimension d is adjusted accordingly and h_d^{t+1} gets re-initialized to 0.5. In this model, different dimensions



(a) EDPSO (b) EDA-PSO (c) PSO_Bounds

Fig. 1 Probabilistic models

Algorithm 4. The PSO_Bounds algorithm

Require: Max_Function_Evaluations, $h_{dmin}, h_{dmax}, \alpha$
Initialize the swarm
Max_Iterations = $\frac{Max\_Function\_Evaluations}{Num\_Particles}$
iter_number = 1
while $iter\_number \leq Max\_Iterations$ **do**
  Update the swarm
  for each dimension d **do**
    $p_d = 0$
    for every particle i **do**
      Calculate p_{id}
      $p_d = p_d + p_{id}$
    end for
    $h_d = (1 - \alpha)h_d + \alpha p_d$
    if $h_d < h_{dmin}$ **then**
      $x_{dmax} = b = \frac{a+b}{2}$
      Update v_{dmin} and v_{dmax}
      $h_d = 0.5$
    else if $h_d > h_{dmax}$ **then**
      $x_{dmin} = a = \frac{a+b}{2}$
      Update v_{dmin} and v_{dmax}
      $h_d = 0.5$
    end if
  end for
  iter_number = iter_number + 1
end while
return *gbest*

might end up having different domains and different velocity bounds which does not happen in normal PSO.

Figure 1 illustrates the approaches taken by the different PSO and EDAs hybridization techniques in order to model the distribution of good solutions across the search space in every dimension.

The steps taken by PSO_Bounds is shown in Algorithm 4, where x_{dmin} and x_{dmax} refer to the minimum and maximum search bounds for dimension d while v_{dmin} and v_{dmax} refer to the minimum and maximum velocity bounds.

6 Results and Discussions

6.1 Experimental Settings

Table 1 shows the parameter settings used for applying the algorithms under study. For all experiments, all the particles have been randomly initialized in the specified domain using uniform distribution. The values for q and ξ are

Table 1 Parameter settings

Model	Parameter	Value
Normal PSO	w	0.9 to 0.1
	c1 and c2	2
EDPSO	q	0.1
	ξ	0.85
EDA-PSO	p	Adaptive - all historical information
PSO_Bounds	α	0.1
	h_{dmin}	0.2
	h_{dmax}	0.8

the same as was proposed in [5] and the value for p is set adaptively using the $allhistoricalinformation$ approach, as it was found to be the best one based on our experiments. The values for (α, h_{dmin}, h_{dmax}) are changed from (0.01, 0.1, 0.9) in [10] to (0.1, 0.2, 0.8) to allow a faster process of varying the bounds. The experiments are conducted for a problem dimensionality of 10, 30, and 50 with 40 particles in the swarm performing 100000, 100000, and 200000 function evaluations, respectively. The results reported are the averages taken over 30 runs.

The experiments are run using the benchmark test functions shown in Table 2.

The experiments are also conducted using the benchmark functions f6-f14 proposed in CEC2005, available at [16] and shown in Table 3. In order to constrain the particles movement within the specified domain for the CEC05 functions, any violating particle gets its position randomly re-initialized inside the specified domain. The error values $f(x) - f(x*)$ are reported, where $x*$ is the global optimum.

In [6], the values for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are calculated using the best half of the swarm. The authors in [17] proposed calculating $\boldsymbol{\sigma}$ using the whole population instead, which is found to produce better results due to the induced diversity avoiding premature convergence. The same approach is used in this work when applying the EDA-PSO algorithm.

Table 2 Benchmark functions

Function	Equation	Domain
Spherical	$f(x) = \sum_{i=1}^{n} x_i^2$	100
Rosenbrock	$f(x) = \sum_{i=1}^{n/2} \left(100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2\right)$	2.048
Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	600
Ackley	$f(x) = 20 + e - 20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$	
	$- \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i\right)$	30
Rastrigin	$f(x) \sum_{i=1}^{n} \left(x_i^2 - 10\cos 2\pi x_i + 10\right)$	5.12

Table 3 CEC05 Benchmark Functions

Benchmark Function	Description	Lower Domain	Upper Domain
f6	shifted Rosenbrock	-100	100
f7	shifted rotated Griewank	0	600
f9	shifted Rastrigin	-5	5
f10	shifted rotated Rastrigin	-5	5
f11	shifted rotated Weierstrass	-0.5	0.5
f12	Schwefel	-100	100
f13	expanded extended Griewank plus Rosenbrock	-3	1
f14	shifted rotated expanded Scaffer	-100	100

Table 4 Results of all the algorithms for the classical functions

Function	Dim.	EDPSO		EDA-PSO		PSO_Bounds	
		Mean	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
Spherical	10	**9.881e-324**	**0**	8.400e-266	0	5.087e-03	2.786e-02
Rosenbrock	10	**5.519e-06**	**1.044e-05**	7.827e-02	8.422e-02	7.744e-01	5.857e-01
Griewank	10	2.084e-02	1.447e-02	**7.882e-03**	**7.325e-03**	1.229e-01	5.988e-02
Ackley	10	**5.887e-16**	**2.006e-31**	1.268e+00	2.258e-15	8.606e-02	4.708e-01
Rastrigin	10	**3.051e+00**	**1.609e+00**	4.013e+00	1.998e+00	7.131e+00	2.172e+00
Spherical	30	3.698e-67	2.026e-66	**4.234e-141**	**1.425e-140**	5.416e+02	3.674e+02
Rosenbrock	30	**9.562e-01**	**2.042e-01**	**1.123e+00**	**4.552e-01**	1.707e+01	4.633e+00
Griewank	30	1.479e-03	3.462e-03	**0**	**0**	4.871e+00	2.021e+00
Ackley	30	**4.378e-015**	**9.014e-016**	1.586e+00	9.034e-16	5.467e+00	1.137e+00
Rastrigin	30	**1.791e+01**	**4.222e+00**	3.4067e+01	2.922e+01	6.799e+01	1.339e+01
Spherical	50	1.104e-59	3.644e-59	**2.811e-103**	**1.539e-102**	2.979e+03	1.131e+03
Rosenbrock	50	**2.078e+00**	**3.954e-01**	**1.565e+00**	**2.745e+00**	3.131e+01	7.791e+00
Griewank	50	**3.286e-04**	**1.800e-03**	2.132e-03	6.314e-03	2.697e+01	8.899e+00
Ackley	50	**7.694e-15**	**1.319e-15**	1.641e+00	2.258e-16	9.068e+00	9.036e-01
Rastrigin	50	**4.016e+01**	**8.593e+00**	**4.630e+01**	**1.410e+01**	1.457e+02	1.891e+01

The best results highlighted in bold in all the tables are selected based on a two-sample t-test where the null hypothesis is rejected with a 95% confidence level.

6.2 Experimental Results

Table 4 shows the results obtained by applying EDPSO, EDA-PSO and PSO_Bounds to the classical functions for different problem sizes.

As shown in Tables 4 for the classical functions, both EDPSO and EDA-PSO outperform PSO_Bounds. The reason for this is that the global optimum is at the center of the search space and the Gaussian model adopted by these algorithms along with the uniform distribution used in initializing the particles make it very easy for these algorithms to reach better results.

Table 5 Results of all the algorithms for the CEC05 benchmark functions

Function	Dim.	EDPSO		EDA-PSO		PSO_Bounds	
		Mean.	Std.	Mean	Std.	Mean	Std.
f6	10	1.375e+00	4.557e+00	**1.123e-02**	**1.626e-02**	1.451e+02	2.218e+02
f7		2.687e-01	2.258e-01	**1.927e-01**	**1.905e-01**	-	-
f9		**3.217e+00**	**1.604e+00**	4.046e+00	2.277e+00	**3.454e+00**	**1.471e+00**
f10		1.989e+01	6.327e+00	**4.819e+00**	**3.642e+00**	7.543e+00	4.528e+00
f11		3.868e+00	3.859e+00	6.588e+00	1.340e+00	**3.529e+00**	**1.730e+00**
f12		2.919e+04	7.054e+03	1.616e+04	6.334e+03	**4.243e+03**	**5.001e+03**
f13		1.194e+00	5.372e-01	8.465e-01	3.968e-01	**6.904e-01**	**1.770e-01**
f14		2.429e+00	5.255e-01	2.667e+00	5.991e-01	**2.365e+00**	**5.792e-01**
f6	30	7.522e+01	1.007e+02	**1.716e+01**	**2.011e+01**	6.602e+05	1.841e+06
f7		**8.700e-03**	**5.920e-03**	1.300e-02	7.589e-03	-	-
f9		**1.175e+00**	**2.044e+00**	2.789e+01	6.498e+00	3.315e+01	7.072e+00
f10		1.850e+02	1.348e+01	1.187e+02	6.191e+01	**5.556e+01**	**2.068e+01**
f11		4.028e+01	1.676e+00	3.494e+01	2.674e+00	**2.849e+01**	**3.897e+00**
f12		1.129e+06	1.266e+05	9.219e+05	2.060e+05	**2.941e+05**	**2.155e+05**
f13		1.489e+01	1.497e+00	7.942e+00	4.688e+00	**4.333e+00**	**7.852e-01**
f14		1.334e+01	2.309e-01	1.325e+01	2.933e-01	**1.245e+01**	**6.541e-01**
f6	50	1.429e+02	2.023e+02	**3.725e+01**	**4.515e+01**	3.458e+07	4.913e+07
f7		**3.000e-03**	**5.813e-03**	9.867e-03	1.374e-02	-	-
f9		**1.282e+01**	**6.519e+00**	4.232e+01	1.080e+01	7.047e+01	1.338e+01
f10		3.765e+02	1.520e+01	2.931e+02	8.820e+01	**1.222e+02**	**2.553e+01**
f11		7.393e+01	1.266e+00	6.744e+01	3.031e+00	**5.778e+02**	**6.800e+00**
f12		5.760e+06	3.738e+05	3.965e+06	1.259e+06	**1.254e+05**	**1.167e+05**
f13		3.057e+01	2.701e+00	1.696e+01	1.109e+01	**9.327e+00**	**2.030e+00**
f14		2.310e+01	2.551e-01	2.282e+01	3.451e-01	**2.237e+01**	**4.455e-01**

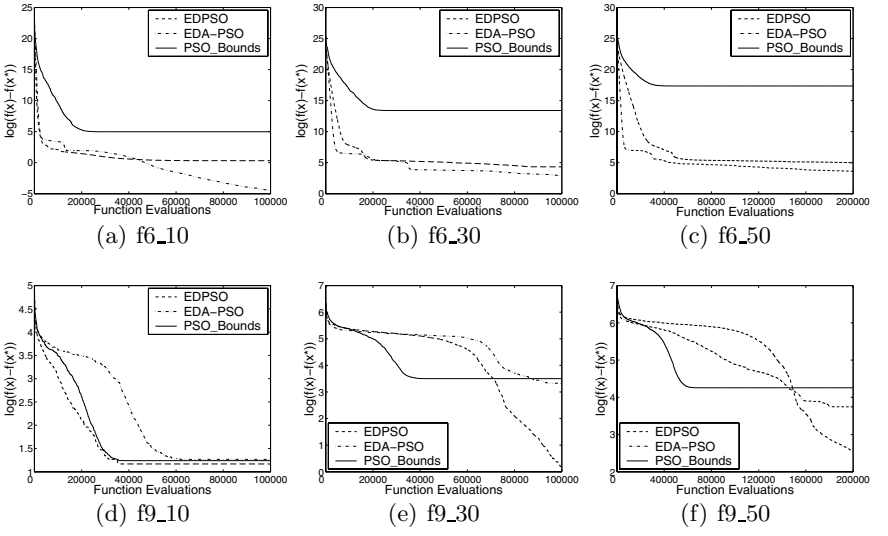
Table 6 Comparison between all the algorithms using the *gbest* model

Algorithm	Classical Functions		CEC05 Functions		Total Number of Cases
	No. of Cases	Best in	No. of Cases	Best in	
PSO_Bounds	-	-	15	f11, f12 f13, f14	15
EDA-PSO	7	-	5	f6	12
EDPSO	11	Rosenbrock Ackley, Rastrigin	5	-	16

On the other hand, for the more difficult CEC05 benchmark functions shown in Table 5, PSO_Bounds has the best performance across the different problem sizes.

Table 6 summarizes the comparison between all the algorithms based on the results shown in Tables 4 and 5. The upper bound for the number of cases is 15 (5 functions in 3 problem sizes) in the classical functions and 21 (7 functions in 3 problem sizes) in the CEC05 functions.

Please note that PSO_Bounds is not applied for f7 as this function is not bounded by a specified domain (the bounds shown in Table 3 are only used as an initialization range).



(a) f6_10 (b) f6_30 (c) f6_50

(d) f9_10 (e) f9_30 (f) f9_50

Fig. 2 Convergence behavior of all the algorithms for the CEC05 functions

The convergence behavior shown in Figure 2 and Figure 3 illustrates that PSO_Bounds usually has a slow speed of convergence compared with the other algorithms. It only has the fastest speed of convergence in both f6 and f9 where it does not produce good results.

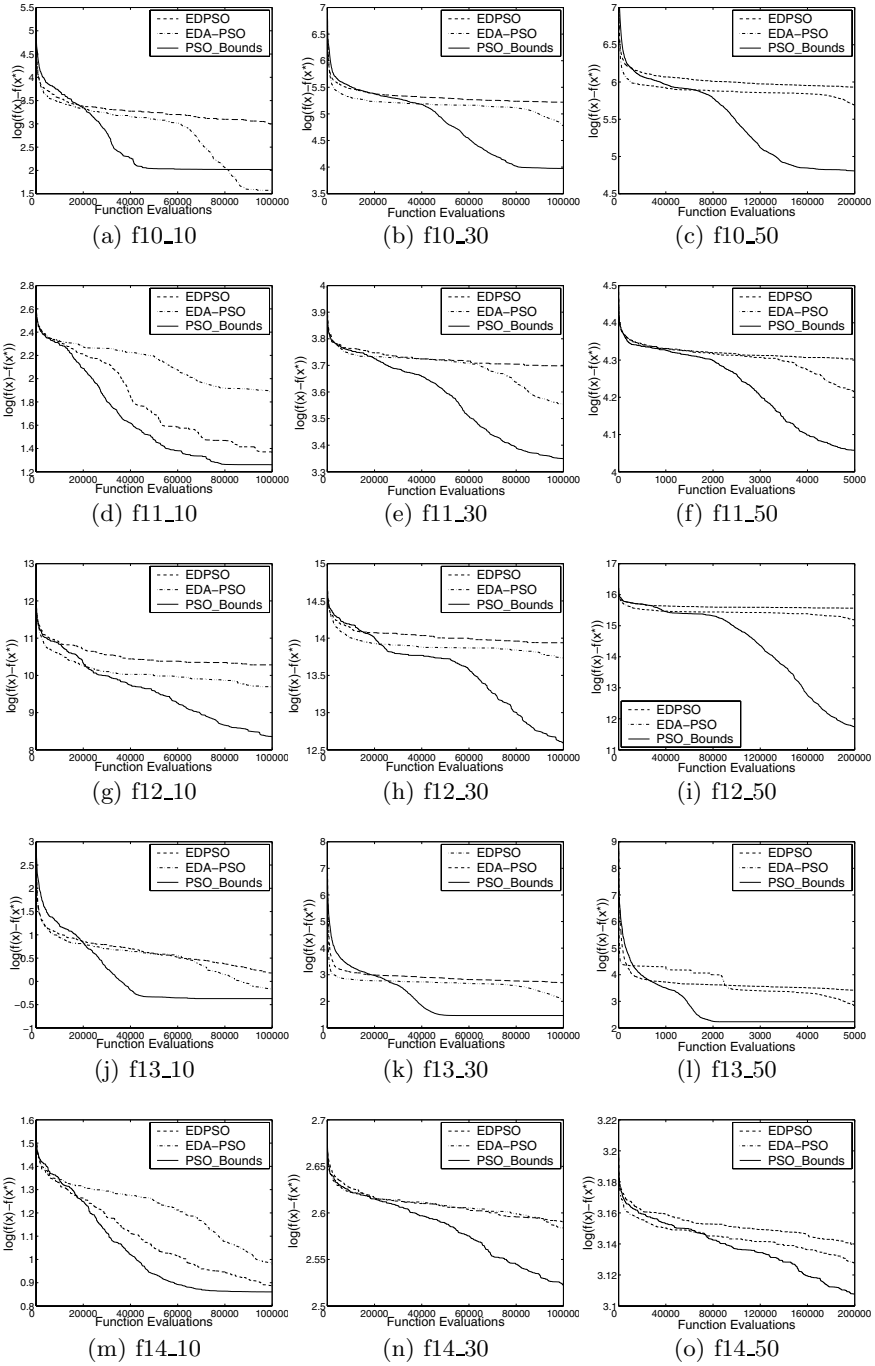
Convergence figures also show that both EDPSO and EDA-PSO have a very similar behavior on most of the functions. This could be due to the fact that both algorithms use the same Gaussian model for sampling the search space.

6.3 Changing the Population Topology

In [18], the authors stated that "modern research performed using only swarms with a global topology is incomplete at best". For this reason, the experiments are rerun again for all the algorithms using the *lbest* model. Table 7 and Table 8 show the obtained results.

The results show that PSO_Bounds still has a deteriorated performance in the classical functions while outperforming other algorithms on the more difficult multimodal functions. This means that all the algorithms exhibit the same performance compared to each other as in the case of using the *gbest* model.

Table 9 summarizes the comparison between all the algorithms based on the results shown in Tables 7 and 8. The results emphasize that the performance of these algorithms (compared to each other) is the same regardless of the underlying population topology.



(a) f10_10 (b) f10_30 (c) f10_50

(d) f11_10 (e) f11_30 (f) f11_50

(g) f12_10 (h) f12_30 (i) f12_50

(j) f13_10 (k) f13_30 (l) f13_50

(m) f14_10 (n) f14_30 (o) f14_50

Fig. 3 Convergence behavior of all the algorithms for the CEC05 functions, contd

Table 7 Results of all the algorithms using the *lbest* model for the classical functions

Function	Dim.	EDPSO_L		EDA-PSO_L		PSO_Bounds_L	
		Mean	Std.	Mean	Std.	Mean	Std.
Spherical	10	**0**	**0**	**3.850e-267**	**0**	**3.194e-17**	**1.722e-16**
Rosenbrock	10	**4.019e-03**	**5.097e-03**	1.029e-01	5.276e-02	2.390e-01	1.781e-01
Griewank	10	1.682e-02	1.194e-02	**4.959e-03**	**7.767e-03**	4.881e-02	1.517e-02
Ackley	10	**5.887e-15**	**2.006e-31**	1.268e+00	2.258e-16	5.037e-11	9.260e-11
Rastrigin	10	**3.118e+00**	**1.5180e+00**	**3.263e+00**	**1.885e+00**	**3.798e+00**	**1.444e+00**
Spherical	30	**6.031e-94**	**3.205e-95**	**7.020e-141**	**9.280e-141**	5.183e-01	1.811
Rosenbrock	30	**1.076e+00**	**1.779e-01**	**1.742e+00**	**2.524e+00**	1.085e+01	3.416e+00
Griewank	30	**2.052e-03**	**4.970e-03**	3.288e-02	1.801e-03	7.657e-02	8.055e-02
Ackley	30	**4.141e-015**	**0**	1.586	3.651e-16	6.166e-01	7.217e-01
Rastrigin	30	**1.523e+01**	**3.999e+00**	4.472e+01	3.057e+01	4.192e+01	7.900e+00
Spherical	50	**3.055e-82**	**1.256e-81**	**3.700e-11**	**2.026-10**	14.233	33.711
Rosenbrock	50	**2.104e+00**	**2.490e-01**	6.237e+00	1.035e+01	2.356e+01	4.209e+00
Griewank	50	**1.232e-03**	**3.284e-03**	**2.919e-03**	**1.465e-02**	6.929e-01	3.780e-01
Ackley	50	**6.865e-15**	**1.528e-15**	1.641e+00	2.258e-16	2.200e+00	6.118e-01
Rastrigin	50	**3.270e+01**	**7.202e+00**	7.481e+01	5.061e+01	9.140e+01	1.469e+01

Table 8 Results of all the algorithms using the *lbest* model for the CEC05 benchmark functions

Function	Dim.	EDPSO_L		EDA-PSO_L		PSO_Bounds_L	
		Mean	Std.	Mean	Std.	Mean	Std.
f6	10	6.554e+00	1.936e+01	**2.092e-01**	**7.899e-01**	1.497e+01	25.785
f9	10	2.919e+00	1.566e+00	3.310e+00	1.259e+00	**8.025e-01**	**9.603e-01**
f10	10	1.946e+01	6.939e+00	1.105e+01	5.716e+00	**6.712e+00**	**3.003e+00**
f11	10	7.292e+00	3.473e+00	6.196e+00	8.1308e-1	**4.480e+00**	**1.027e+00**
f12	10	2.729e+04	7.468e+03	1.877e+04	6.712e+3	**6.535e+03**	**2.841e+03**
f13	10	1.435e00	4.549e-01	1.224e+00	3.724e-01	**6.422e-01**	**1.390e-01**
f14	10	**2.204e+00**	**5.145e-01**	**2.910e+00**	**2.684e-01**	**2.777e+00**	**3.261e-01**
f6	30	**8.592e+01**	**1.305e+02**	**7.063e+01**	**4.586e+01**	8.883e+03	3.632e+04
f9	30	**1.605e+01**	**5.372e+00**	4.208e+01	2.742e+01	2.536e+01	4.694e+00
f10	30	1.778e+02	9.953e+00	1.608e+02	1.719e+01	**1.384e+02**	**1.864e+01**
f11	30	4.043e+01	1.148e+00	3.641e+01	2.107e+00	**3.163e+01**	**2.479e+00**
f12	30	1.140e+06	1.148e+05	9.571e+05	1.696e+05	**4.978e+05**	**1.443e+05**
f13	30	1.447e+01	1.328e+00	1.156e+01	2.639e+00	**4.755e+00**	**9.558e-01**
f14	30	1.347e+01	1.873e-01	1.327e+01	2.282e-01	**1.302e+01**	**2.674e-01**
f6	50	**6.550e+01**	**5.446e+01**	**6.789e+01**	**4.228e+01**	1.967e+06	1.012e+07
f9	50	**3.250e+01**	**6.450e+01**	5.334e+01	2.326e+01	5.547e+01	9.813e+00
f10	50	3.629e+02	1.624e+01	3.359e+02	1.660e+01	**2.879e+02**	**4.048e+01**
f11	50	7.381e+01	1.911e+00	6.926e+01	2.552e+00	**6.184e+01**	**4.231e+00**
f12	50	5.631e+06	4.676e+05	4.725e+06	5.695e+05	**1.896e+06**	**3.675e+05**
f13	50	2.738e+01	4.029e+00	2.446e+01	4.217e+00	**1.062e+01**	**2.183e+00**
f14	50	**2.316e+01**	**1.693e-01**	**2.292e+01**	**2.184e-01**	**2.261e+01**	**2.103e-01**

Table 9 Comparison between all the algorithms using the *lbest* model

Algorithm	Classical Functions		CEC05 Functions		Total Number of Cases
	No. of Cases	Best in	No. of Cases	Best in	
PSO_Bounds	2	-	16	f10, f11 f12, f13	18
EDA-PSO	7	Spherical	5	f6	12
EDPSO	14	Spherical, Rosenbrock Ackley, Rastrigin	6	-	20

7 Conclusion and Discussion

This chapter gives a brief introduction to Particle Swarm Optimization and Estimation of Distribution Algorithms (EDAs). The chapter surveys the different methods previously adopted to combine PSO and EDAs.

The chapter introduces a new algorithm, PSO_Bounds, which is a PSO algorithm that borrows ideas from PBIL. The new algorithm uses the same equations of motion as PSO while using the current distribution of the particles during the search to continuously update the allowable search domain.

Along with the proposed algorithm, all the approaches covered are implemented and compared using a suite of well-known benchmark optimization functions with different properties. It is shown that PSO_Bounds outperforms other PSO and EDAs hybridization techniques on the more difficult shifted and/or rotated multimodal functions. It is also shown that the new proposed algorithm has in general a slower speed of convergence when compared to other algorithms.

Moreover, the relative performance of all the algorithms is shown to be independent of the underlying population topology used by the PSO component.

Many future directions could be followed to further improve on the performance of such algorithms. The deteriorated performance of PSO_Bounds in some functions could be due to the fact that the width of the allowable domain for the different dimensions becomes smaller and smaller as the search progresses. It would eventually get to the point of being very close to zero (or zero, even). Once this happens, the particles will stop moving as the allowable movement domain for the particles is very small as well as the allowable maximum velocity, hence, the search stagnates. One way to improve this is to re-initialize those domains again, by re-setting them to the initial search ranges, if the width drops under a pre-determined threshold.

A similar approach could be adopted for EDA-PSO by re-initializing the current positions of the particles, while keeping their *pbests* values as they are so as not to lose any useful information, if the value of σ drops under a pre-determined threshold during the search.

A different research direction is to incorporate PSO with probabilistic models that allow inter-variable dependencies. All the hybridization techniques

proposed up-to-date, including the one in this chapter, use probabilistic models that assume that the problem variables are independent.

References

1. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
2. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. of the 6th International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
3. Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer, Dordrecht (2002)
4. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. School of Computer Science, Carnegie Mellon University, Tech. Rep. CMU-CS-94-163 (1994)
5. Iqbal, M., de Oca, M.A.M.: An estimation of distribution particle swarm optimization algorithm. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) Proc. of the Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence, pp. 72–83 (2006)
6. Zhou, Y., Jin, J.: Eda-pso - a new hybrid intelligent optimization algorithm. In: Proc. of the Michigan University Graduate Student Symposium (2006)
7. Eberhart, R.C., Simpson, P., Dobbins, R.: Computational Intelligence.PC Tools: Academic, ch. 6, pp. 212–226 (1996)
8. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proc. of IEEE Congress on Evolutionary Computation, vol. 2, pp. 1671–1676 (2002)
9. Rudolph, S., Koppen, M.: Stochastic hill climbing with learning by vectors of normal distributions. In: First on-line Workshop on Soft Computing (WSC1), pp. 60–70 (1996)
10. Servet, I., Trave-Massuyes, L., Stern, D.: Telephone network traffic overloading diagnosis and evolutionary computation technique. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D. (eds.) AE 1997. LNCS, vol. 1363, pp. 137–144. Springer, Heidelberg (1998)
11. Sebag, M., Ducoulombier, A.: Extending population-based incremental learning to continuous search spaces. In: Proc. of Parallel Problem Solving from Nature, pp. 418–427 (1999)
12. Gallagher, M., Frean, M., Downs, T.: Real-valued evolutionary optimization using a flexible probability density estimator. In: Proc. of Genetic and Evolutionary Computation Conference, vol. 1, pp. 840–846 (1999)
13. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications 21(1), 5–20 (2002)
14. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. Universitie Libre de Bruxelles, Tech. Rep. TR/IRIDIA/2005-037 (2005)
15. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operationl Research 185(3), 1155–1173 (2008)

16. CEC05 benchmark functions,
http://staffx.webstore.ntu.edu.sg/MySite/Public.aspx?accountname=epnsugan
17. delaOssa, L., Gamez, J., Puerta, J.: Initial approaches to the application of island-based parallel edas in continuous domains. Journal of Parallel and Distributed Computing 66(8), 991–1001 (2006)
18. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Proc. IEEE Swarm Intelligence Symposium, pp. 120–127 (2007)