Particle Swarm Optimization in Linear Optimization Problems

Nuria Gómez Blas, Luis Fernando de Mingo López, Juan Castellanos Peñuela

Abstract: *Swarm colonies reproduce social habits. Working together in a group to reach a predefined goal is a social behaviour occuring in nature. Linear optimization problems have been approached by different techniques based on natural models. In particular, Particles Swarm optimization is a meta-heuristic search technique that has proven to be effective when dealing with complex optimization problems. This paper presents and develops a new method based on different penalties strategies to solve complex problems. It focuses on the constraints and the election of the parameters to ensure successful results.*

Keywords: *Partile Swarm Optimization, Non Linear Optimization, Artificial Intelligence, Swarm Intelligence.*

ACM Classification Keywords*: F.1.1 Theory of Computation - Models of Computation, I.2.6 Artificial Intelligence.*

MSC*: 68Q32 Computational learning theory, 68T05 Learning and adaptive systems.*

Introduction

Particles Optimization is a technique based on heuristics, which emulates the intelligence of social and biological organizations. It is mainly used when dealing with optimization problems. Swarms of individuals are moving freely within an area searching for food; when an individual finds it somehow communicates with its neighbors, so that the swarm will follow the individual. A model of individuals called particles was originally designed to optimize functions in a binary search space. Then, this model was modified for problems of unrestricted searching spaces with continuous variables. Currently it is being tested for applications of nonlinear optimization problems with constraints and multitarget problems. Moreover, there are some models which have been developed to use hybrid techniques combined with genetic algorithms and fuzzy logic, neural networks and others.

This paper presents a standard problem of optimization of particles and some of its variants, emphasizing the problem of nonlinear constrained optimization. It also uses different methods of penalty. This work shows examples of PSO algorithms implemented in C ++.

Particles optimization model

Particle swarm optimization (PSO) is a heuristic optimization technique, originally developed by James Kennedy and Russell C. Eberhart in 1995. PSO is considered to fall within the branch of evolutionary computing, which is commonly referred as swarm intelligence. Evolutionary Computation is a collection of methods that simulate evolution by using computers, including genetic programming, evolution strategies, Swarm Optimization, Artificial Life and other related fields. Evolutionary computation and genetic algorithms share the technique for generating populations in a stochastic way. Furthermore, this technique is used to generate new generations too.

The algorithm used by PSO is inspired by the collective work of swarms, specifically in the social organization of bird flocks and fish schools. These social organizations take on a collective behavior that comes from communication and cooperation among its members. In the algorithm, individuals of the population are treated either as particles or searching points, which have a position and velocity (the velocity vector indicates where it is headed). Then they move into an area of dimension n. The particle that gets the best assessment is the *leader*, in the sense that any particle in the swarm follows it. However any individual could change their orientation. In that case the leader passes leadership to the one that changed the orientation. The *leader* can either be global leader in the whole swarm or local in a part of it.

The emulation of these organized societies has been successful in discrete and continuous optimization problems. Initially, Kennedy and Eberhart developed a standard algorithm, which has undergone several improvements. This is a fairly simple algorithm and widely used today.

Particles move along the search space by using a combination of the best individual solution found by the particle and the best found by any of the neighboring particles. Performance of every particle is monitored.

The three main operations Kennedy and Eberhart [9] algorithms use are: To assess, to compare and to imitate. Evaluation is one of the most important features of some living organisms; they evaluate, learn and evolve. Besides, organisms analyze their neighbors and imitate only those who have better performance. In general, these three operations can be applied to simple social beings and to computer programs. This enable them for solving complex problems.

Considering that partciles reside in a space of dimension d , each particle i is related to three vectors: position: $x^(i)) = (x_1, x_2 \ldots, x_d)$, the best position of its history: $p^(i)) = (p_1, p_2 \ldots, p_d)$ and speed $v^(i)) = (v_1, v_2 \ldots, v_d)$. Initially values related to particles are generated randomly, then these particles move around a search space by using a system of equations that is updated for every iteration. The intention is to find the best solution. Each particle moves to the more succesful neighbor, influencing the other particles. The algorithm updates the swarm at each step. It also changes the position and velocity of each particle and it applies the following rules:

$$v_d^{(i)} = v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \quad (1)$$

$$x_d^{(i)} = x_d^{(i)} + v_d^{(i)} \quad (2)$$

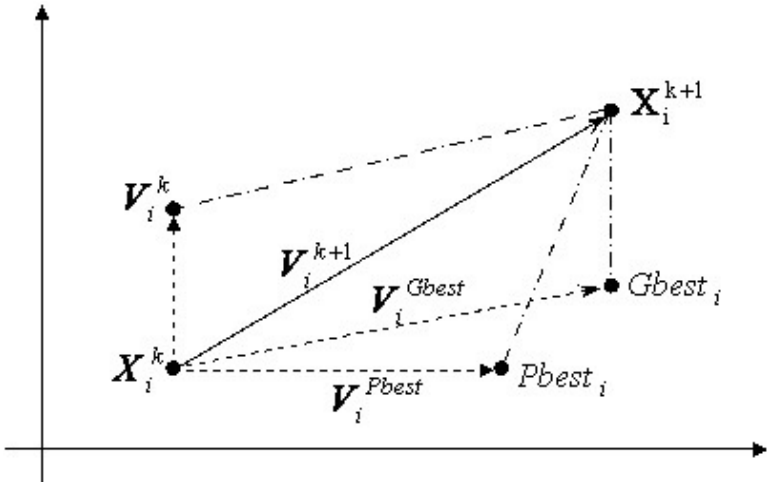

Figure 1: Original models of particles relations.

where $v_d^{(i)}$ is the d-th component of the velocity of particle i, $x_d^{(i)}$, the d-th component of vector position of the particle i, c_j, $(j = 1, 2)$ is a constant value (obtained from experimental results) and ϵ_1 and ϵ_2 are independent random numbers uniformly distributed in $[0.1]$. They are generated for every update, p_d is the d-th component of the improved performance of the particle in its history, and g_d is the d-th component of the particle with the best position found between neighboring particles, throughout its history. This neighborhood is defined according to the topology of the particle system. The factor c_1 is known as the factor of personal or cognitive learning and the factor c_2 as the social learning factor. Both factors have much influence on the rate of convergence of the optimization process.

Mathematical models developed for PSO vary according to how the particles interact with their neighbors, this is known as the topology of the system, which can be understood as the way the swarm of particles organizes itself. Early models of PSO used a Euclidean neighborhood. However the number of operations were high; in order to reduce the number of operations, a new mathematical model was developed. In this mathematical model neighborhoods were not related to the location of the particle; these are called local neighborhoods or *lbest models*. They can be also global or and *gbest models*, see figure (1). Originally *gbest* had better performance, but recent research has also shown good results in some problems when using the model *lbest* by adding some improvements to the algorithm.

In PSO, neighborhood is understood as the set of particles related to a given particle. This relationship influences the search capability and convergence. In the ring-type topology, each particle communicates only with n neighbors, $n/2$ on each side. Figure 2, the ring topology presented is the simplest: $n = 2$. This means that only two neighboring particles are evaluated. In the wheel topology all information is concentrated in a central particle. In the star type each particle is related to all the particles of the swarm.

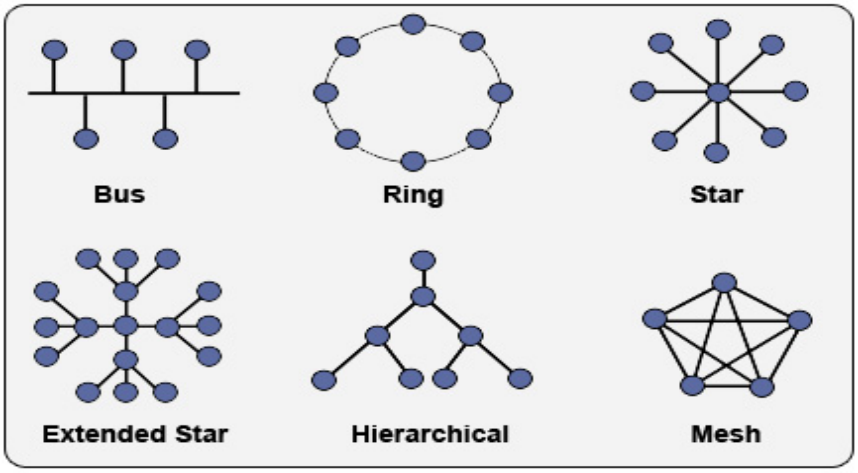

Figure 2: Different topologies.

The standard PSO algorithm is described as follows:

Algorithm

BEGIN

Create initial population of particles $\{x^i\}$

FOR each particle of the swarm DO

IF $f(x) > f(p)$ THEN DO $//f$ *fitness funtcion*

FOR $d = 1$ TO D DO

$p_d = x_d$ $//p_d$ *It is the best so far*

ENDFOR

ENDDO

$g = i$

FOR $j \in J$ $//J$ *Set of indexes of neighboring particles*

IF $f(p_j) > f(p_g)$ THEN $g = j$ $//g$ *is the index of the best particle in the neigborhood*

ENDFOR

FOR $d = 1$ TO D Do

$$v_d(t) = v_d(t-1) + c_1\epsilon_1(p_d - x_d(t-1)) + c_2\epsilon_2(g_d - x_d(t-1)) \quad [1]$$

$x_d = x_d + v_d$

ENDFOR

[1] $v_d \in (-V_{max}, +V_{max})$

ENDFOR

END

It begins with a population of particles with position and velocity randomly assigned in the search space. With regard to the velocity of the particles, we consider a maximum value as restriction. This is called V_{max}. This is done because an explosion might happen, since the velocities could be increased quickly. The selection of the values V_{max} is difficult to determine. They will be chosen by trial and error. In very large spaces, large values are usually selected in order to ensure proper exploration. This is justified since a large inertia weight facilitates exploration in new areas in the global search space. On the other hand a small one facilitate the exploration in a local area. In the case of small spaces, small values are required to prevent the explosion. As for the size of the particle population, empirical results have shown that good results are not always obtained by increasing the number of particles of the population . Some examples have been influenced by that but others have not.

01 Model variants

Regarding the PSO algorithm, different variants have been developed. Most of them aimed at speeding up the convergence of it. In addition to the unconstrained optimization problem in discrete or continuous variable, the multitarget problem and the constrained problem have been addressed. We have also developed some hybrid optimization techniques. PSO technique has been tested with good results for training Artificial Neural Networks. When applying the method of Back Propagation, we are able to find appropriate weights that minimize an error function through a succession of iterations. Furthermore, by applying the PSO technique, the weights found are more efficient just by making small modifications to the algorithm. The new guidelines are aimed at avoiding PSO stagnation of the local optimal solutions.

Shi and Eberhart [13] proposed adjustments to the velocities of the particles by using a factor w called *inertial weight*. This factor utilizes the inertia of the particles in the process of friction when they are moving. This modification in the algorithm is done to control the search space. In order to do that it must change (3). The large inertia weight makes the global search easier; however small inertia weight does not improve local search. That is why was the initial value is greater than 1.0 to promote global exploration, and then gradually decreases to obtain more refined solutions. The algorithm decreases linearly at each iteration. Moreover, the use of inertial weight removes the restriction V_{max} on the velocity.

$$v_d^{(i)} = wv_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \tag{3}$$

In each iteration, inertia weight decrease linearly through the following expression:

$$w = w_{max} - (w_{max} - w_{min})\frac{g}{G} \tag{4}$$

g is the index of the generation, G is the maximum number of iterations previously determined, w_{max} is a value greater than 1, and w_{min} a value under 0.5. This variation of the method has proven to accelerate convergence.

Clerc and Kennedy [4] obtain another variation in the speed calculation. A constriction factor χ is introduced with that purpose, This factor depends on the constants that are used when calculating speed and it affects to the formula (1) The aim is to avoid the explosion of velocity:

$$v_d^{(i)} = \chi[v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)})] \tag{5}$$

χ is:

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \quad \varphi = c_1 + c_2 = 4.1$$

The results are: $\chi = 0.729$ and c_1=c_2= 2.05. These parameters were obtained by performing several tests.

χ factor is similar to the inertial weight. This means that controlling the velocity with $V_(max)$ is not required when χ is used. Bratton and Kennedy [2], analyzed the stability of this algorithm by using these values and by following a comparative study of both PSO algorithms (inertial weight and χ factor). Both of them are mathematically equivalent, in particular the algorithm with constriction factor is a special case of the inertial weight. Moreover, Parsopoulos al [12], combined both for problems with constraints and they obtained equally good results in several tests.

We observe that the convergence always becomes slower when problem size increase, so when it comes to high-dimensional problems, a larger number of iterations occurs. Researchers Hatanaka et al [6] developed a PSO model, where velocity values are updated, by considering the rotation of the coordinate system. This model is aimed at problems of high dimensionality and it showed good results when applied to all functions of De Jong, (larger dimensions).

Numerical Examples

In order to test the standard PSO algorithm and two variants with incorporated and inertial weight factor χ, we have used some unrestricted functions which are commonly referred as *De Jong functions*'. The minimum of these functions is located in the search space. They were originally proposed by de Jong to measure the performance of genetic algorithms. However they have also been used to test the performance in PSO algorithms. Some of the other functions are unimodal and multimodal i.e Ring (*lbest*) and star (*gbest*) topologies. In the ring topology each particle is related to its two neighbors. In the star topology all particles are interconnected. A population of 20 particles was considered. Table 1 functions are tested with the standard PSO algorithm and two variations: inertia weight and constriction factor. The first three functions are unimodal and have the optimal solution $x^* = 0.0^d$ and the minimum value $f_i(x^*) = 0.0$. The following are multimodal functions, the function f_4 has the minimum value 0.0, the optimal solution is $\pm n\frac{\pi}{2}^d$, the function f_5 has optimal solution $x^* = 0.0^d$ and the minimum value of the function: $f(x^*) = 0.0$ and f_6 has the optimal solution $x^* = 420.968^d$ and the minimum value of the function : $f(x^*) = -12.569, 4866.$

Tables (2) and (3) show the results after 20 executions of the standard algorithm. Not only the inertial weight has been modified in this algorithm but also the constriction factor for each functions by using neighborhood models *lbest* and *gbest*. The algorithm stops when two successive values of the best assessments of the swarm get close to each other. (A ϵ value is prefixed and so it is a maximum number of iterations). NPE

Function	Dim.	Search space.	Name
$f_1(x) = \sum_{i=1}^{d} x_i^2$	30	[-100,100]	Sphere/Parabola
$f_2(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	Rosenbrock Generaliz.
$f_3(x) = \sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	30	[-100,100]	Schwefel 1.2
$f_4(x) = \sum_{i=1}^{d} cos(x_i)^2$	30	$(-\infty, \infty)$	
$f_5(x) = \sum_{i=1}^{d}(x_i^2 - 10cos(2\pi x_i) + 10)$	30	[-5.2, 5.2]	Rastrigin Generaliz.
$f_6(x) = -\sum_{i=1}^{d} x_i \sin(\sqrt{\lvert x_d \rvert})$	30	[-500,500]	Schwefel Generaliz. 2.6

Table 1: Functions tested with PSO algorithm

(average number of assessments) shows the average number of evaluations for the function when applying PSO and its variants.

Name	PSO Original		Peso Inertial		Factor Constriction	
	Best Solut.	NPE	Best Solut.	NPE	Best Solut.	NPE
f_1	**l**: 3,70889e-07	190.480	7,68359e-07	28.800	2,69078e-08	19.100
	g: 5,9803e-04	2×10^5	2,45656e-20	151.500	3,63055e-20	30.820
f_2	**l**: 5,60357e-06	2×10^5	2,2112e-05	831.580	4,5089e-06	1.257.520
	g: 4,51068e-02	4×10^6	9,24376e-12	4×10^6	2,59676e-09	5×10^5
f_3	**l**: 5,76231e-06	198.280	5,57901e-15	1.425.900	2,83076e-16	399.140
	g: 4,91261e-06	2×10^5	1,81786e-12	2.596.780	1,44177e-13	125.640

Table 2: Results obtained by applying the PSO algorithm to unimodal functions, l indicates the model *lbest*; g refers to model *gbest*.

After PSO Algorithm and its variants are executed, results are collected; best results are found in approximately equal number of cases regardless the model is used (*lbest* or *gbest*), so we can not assure which topology is the optimal. Moreover, we notice that when using the constriction factor, the convergence accelerates and the results are better when compared to the exact solution. In some cases, we notice that the region in which the swarm of particles starts can influence the results. Regarding the number of particles of the swarm, when increased to more than 20, results did not improve.

Problems with Constraints

An optimization problem (minimization) with restrictions is defined as follows:

Minimize $\mathbf{f}(\mathbf{x})$

Constrains: $\mathbf{g_j}(\mathbf{x}) \leq \mathbf{0} \quad (j = 1, \ldots, p)$

$\mathbf{h_j}(\mathbf{x}) = \mathbf{0} \quad (j = p+1, \ldots, m)$

$x \in \Re^n$

Name	PSO Original		Weight Inertial		Factor Constriction	
	Best solut.	NPE	Best solut.	NPE	Best solut.	NPE
f_4	**l**: 2.18719e-05	2×10^5	4.36373e-19	114.560	6,7306e-19	4.174
	g: 3,9543e-12	180.820	1.3000e-12	15.120	1.7063e-15	14.160
f_5	**l**: 1,70688e-14	216.240	1,81227e-14	339.020	1,07181e-11	9.480
	g: 3,00262e-04	2×10^5	6,82288e-12	12.180	2,81599e-12	11.040
f_6	**l**: -12.568,2	2×10^5	-12.569,5	2×10^5	-12.569,5	2×10^5
	g: -12.569,5	6×10^5	-12.569,5	6×10^5	12.352,3	2×10^5

Table 3: Results obtained by applying the PSO algorithm to unimodal functions, l indicates the model *lbest*; g refers to model *gbest*.

The problem of nonlinear constrained optimization arises frequently in engineering.In general it does not have a deterministic solution. In the past, nonlinear optimization methods were developed and now it is a challenge to work with differentiable functions.Before gradient methods were used succesfully for solving some problems. Evolutionary methods provide a new possibility for solving such problems. The PSO technique has been used successfully in optimizing real functions without restrictions, but it has been little used for problems with restrictions. This has happened mainly because there are no mechanism to incorporate restrictions on the *fitness* function. Evolutionary Computation has tried to solve the constrained optimization problem, either by bypassing nonfeasible solutions sequences, or by using a penalty function for nonfeasible sequences. Some researchers suggest to use two subfunctions of *fitness*. One helps to evaluate feasible elements and the other one evaluates the unfeasible onea . In this regard, there are many criteria. Moreover, some special self adaptive functions have been designed to implement the penalty technique.

Hu and Eberhart [8] presented a PSO algorithm. This algorithm bypasses nonfeasible sequences. it also creates a random initial population,in which nonfeasible sequences are bypassed until the entire population has only feasible particles. By upgrading the positions of the particles nonfeasible sequences are bypassed automatically. The cost of the technique that creates the initial populations is high, especially when it comes to problems with nonlinear constraints because then it must create an entire population of feasible individuals. In his work, Cagnina et al [3] proposed the following strategies for implementing the PSO into problems with restrictions: **a)** If two particles are feasible, select the one with the best *fitness*. **b)** When one particles is feasible and the other is not, the feasible one is chosen. **c)** If two particles are nonfeasible, the one with the lowest degree of nonfeasibility is selected. These strategies are applied when the particles *gbest* and *lbest* are selected . The same authors also proposed an update in (1). This update considers three elements:

1. $p_d^{(i)}$ which is the best position reached by the particle i in its history,
2. $g_d^{(i)}$ which is the best position reached by the particles in its neigborhood and,
3. t_d which is the best position achieved by any particle in the whole swarm.

$$v_d^{(i)} = w(v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) + c_3\epsilon_3(t_d - x_d^{(i)})) \tag{6}$$

where c_1 is the personal learning factor and c_2 and c_3 are the social learning factors. According to Michalewicz et al [10] y [11] constrained optimization methods are classified as:

1. Methods based on preserving feasibility of solutions.
2. Methods based on penalty functions
3. Methods that make a clear distinction between feasible solutions and infeasible sequences.
4. Methods based on decoders
5. Hybrid methods

Part of our work is to analyze penalty methods under E.A. perspective (Evolutionary Algorithms). The penalty methods use functions (penalty functions) that degrade the quality of the nonfeasible solution. In this way the constrained problem becomes a problem without constraints by using a modified evaluation function:

$$eval(x) = \begin{cases} \text{f(x)} & x \in \mathcal{F} \\ \text{f(x) + } \textit{penalty}\text{(x)} & eoc \end{cases}$$

where $\mathcal{F}$ is the set created by the intersection of all sets that are the restrictions of the problem (Feasible region). The penalty is zero if no violation occurs and it is positive otherwise. The penalty function is based on the distance beteween a nonfeasible sequence and the feasible region $\mathcal{F}$, It also works for repairing solutions outside of the feasible region $\mathcal{F}$.

There are many penalty methods. The main difference between the methods is the way the penalty function is designed and applied to the nonfeasible sequences. Some methods associate a penalty function f_j, $(j = 1, \ldots, m)$ with a constraint, which measures the violation of the restriction j as follows:

$$f_j(x) = \begin{cases} max\{0, g_j(x)\}, & si \;\; 1 \leq j \leq p \\ |h_j(x)| & si \;\; p+1 \leq j \leq m \end{cases}$$

According to Michalewicz al[10], penalty methods are classified as **a)** Static, **b)** Dynamic, **c)** Annealing **d)** Adaptive **e)** death. There are other hybrid methods that combine different techniques. Static methods use l violation levels of constraints; to do that it creates a penalty coefficient $R_{ij}, (i = 1, 2, \ldots, l, j = 1, 2, \ldots, m)$, for each constraint. The greater the violation of the restriction, the greater its value. It begins with a population that may have nonfeasible elements. *fitness* function is as follows:

$$eval(x) = f(x) + \sum_{j=1}^{m} R_{ij} f_j^2(x)$$

The main problem of this methods is the number of parameters involved.

Dynamic penalties method

Individuals are evaluated in each iteration (generation) k, by using the following formula:

$$eval(x) = f(x) + (C.k)^{\alpha} \sum_{j=1}^{m} f_j^{\beta}(x)$$

C, α and β are constant values. The penalty changes over time. A reasonable selection of these parameters is: $C = 0,5$, $\alpha = \beta = 2$. $(Ck)^{\,alpha}$ is larger when k increases. $Functions f_j$ represent restrictions violation. They are calculated as $f_j = max\, o, g_j(x)$

This method obtains the following results when applied to the following set of test functions:

Function	$f(x^*)$	Best solution	Media	Est Deviation.
g01	-15.0	-15.0	-13.86241	1.29431593
g02				
g03				
g04	-30.665,539	-30.687,8	30.992.42	657.818933
g05				
g06	-6961.63	-6962.63	-6961.6725	12.93028533
g07	24,3062091	24,4253	26,18224	1,61696066
g08	-0,095825	-0.095825	-0,09576949	0,00010912
g09	680,6300573	680,649	680.86865	0,22674309
g10				

Restrictions such as 'equal to' are difficult to work with;

$|h_j(x)| \leq \epsilon$,

That is the reason why we decide to relax these constraints by setting a prefixed value to ϵ=10^{-6}

A proper adjustment of parameters plays an important role in PSO, for example the adjustment of maximum speed to the the range of values that the inertial weight can take. Upgrading the position and speed of particles (original equations 1 and 2) is another parameter that plays a major role in PSO. The latter are related to social influence exerted by each particle in the swarm (topology of the particles) and the overall behavior of the swarm. The other parameters apply to the initial generation of particles.

$$v_d^{(i)} = v_d^{(i)} + c_1\epsilon_1(p_d^{(i)} - x_d^{(i)}) + c_2\epsilon_2(g_d^{(i)} - x_d^{(i)}) \tag{7}$$

$$x_d^{(i)} = x_d^{(i)} + v_d^{(i)} \tag{8}$$

The data included in the table are the satisfactory results. We have detected cases that need to be improved. In these cases we re working on the adjustment to the parameters of the particles and to the penalty function. Maurice Clerc(*Stagnation Analysis in PSO*, 2006) proposes amendments to the standard model PSO-2007 Kennedy. PSO has developed several models to improve the classical model by following that way. They are no differences regarding the values of c_1 and c_2. In this regard we have tried some configurations, but the improvements have not been significant. In connection with the neighborhoods, Kennedy and Eberhart (textit Bare Bones Particle Swarm, 2003) have proposed the use of Gaussians neighborhoods. This is an adjustment to the particle position by using an estimation of the average distribution between the best recorded position of the particle ($p^{(i)}$) and the best position within its neighborhood ($g^{(i)}$). The standard deviation is the difference between these two values (see equation ref gauss).This will gain new positions of the particles by using $N(\ mu,\ sigma)$. Some authors compare this method to the random variation of the mutations occuring in Evolutionary Computing because the particle position suffers a random change caused by a different rule.

$$N(\frac{|p^{(i)} - g^{(i)}|}{2}, |p^{(i)} - g^{(i)}|) \tag{9}$$

Moreover Kennedy and Mendes in (*Neighborhood Topologies in Fully Informed and Best Of Neighborhood Particle Swarms*, 2006) have used the Von Neumann neighborhood to deal succesfully with other types of problems. This will be object of our study in further work.

Test Functions

Below is a description of the functions to be tested using the penalty methods:

g01: Minimizing $f(x) = \sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$

constraints:

$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$

$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$

$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$

$g_4(x) = -8x_1 + x_{10} \leq 0$

$g_5(x) = -8x_2 + x_{11} \leq 0$

$g6(\ x) = -8x_3 + x_{12} \leq 0$

$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0$

$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0$

$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0$

$0 \leq x_i \leq 1(i = 1, \ldots, 9), 0 \leq x_i \leq 100$ (i = 10; 11; 12) y $0 \leq x_{13} \leq 1$

Global optimum is : $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, $f(x^*) = -15$.

g02: Maximizing: $f(x) = \left| \frac{\sum_{i=1}^{n} cos^4 x_i - 2\prod_{i=1}^{n} cos^2 x_i}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$

constraints:

$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$

$g_2(x) = \sum_{i=1}^{n} x_i - 7.5 \leq 0$

and $n = 20$ y $0 \leq x_i \leq 10$, $(i = 1 \ldots n)$.

Global optimum is unknown, the best output value for $f(x)$ is $f(x^{\,ast}) = 0.803619$.

g03: Maximizing: $f(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$

constraints:

$h(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$

$n = 10$ y $0 \leq x_i \leq 1, (i = 1, \ldots, n)$.

Local optimum is $x_i^* = 1/\sqrt{n}$, (i=1,...,n), $f(x^*) = 1$.

g04: Minimizing: $f(x) = 5,3578547x_3^2 + 0,8356891x_1x_5 + 37,293239x_1 - 40792,141$

constraints:

$g_1(x) = 85,334407 + 0,0056858x_2x_5 + 0,0006262x_1x_4 - 0,0022053x_3x_5 - 92 \leq 0$

$g_2(x) = -85,334407 - 0,0056858x_2x_5 - 0,0006262x_1x_4 + 0,0022053x_3x_5 \leq 0$

$g_3(x) = 80,51249 + 0,0071317x_2x_5 + 0,0029955x_1x_2 + 0,0021813x_3^2 - 110 \leq 0$

$g_4(x) = -80,51249 - 0,0071317x_2x_5 - 0,0029955x_1x_2 - 0,0021813x_3^2 + 90 \leq 0$

$g_5(x) = 9,300961 + 0,0047026x_3x_5 + 0,0012547x_1x_3 + 0,0019085x_3x_4 - 25 \leq 0$

$g_6(x) = -9,300961 - 0,0047026x_3x_5 - 0,0012547x_1x_3 - 0,0019085x_3x_4 + 20 \leq 0$

$78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45, 27 \leq x_i \leq 45 (i = 3; 4; 5)$.

Optimal solution is: x^* = (78, 33, 29.995256025682, 45, 36.775812905788), $f(x^*)$ = -30665.539

g05: Minimizing: $f(x) = 3x_1 + 0,000001x_1^3 + 2x_2 + (0,000002/3)x_2^3$

constraints:

$g_1(x) = -x_4 + x_3 - 0,55 \leq 0$

$g_2(x) = -x_3 + x_4 - 0,55 \leq 0$

$h_3(x) = 1000sin(-x_3 - 0,25) + 1000sin(-x_4 - 0,25) + 894,8 - x_1 = 0$

$h_4(x) = 1000sin(x_3 - 0,25) + 1000sin(x_3 - x_4 - 0,25) + 894,8 - x_2 = 0$

$h_5(x) = 1000sin(x_4 - 0,25) + 1000sin(x_4 - x_3 - 0,25) + 1294,8 = 0$

and $0 \leq x_1 \leq 1200, 0 \leq x_2 \leq 1200, -0,55 \leq x_3 \leq 0,55, -0,55 \leq x_4 \leq 0,55$

Optimal solution: x^* = (679.9453, 1026.067, 0.1188764, -0.3962336), and $f(x^*)$ = 5126.4981.

g06: minimizing: $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

constraints:

$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$

$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82,81 \leq 0$

and $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$

optimal solution: x^* = (14.095, 0.84296), and $f(x^*)$ = -6961.81388.

g07: minimizing: $f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2$
$+2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$

constraints:

$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$

$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$

$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$

$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 \leq 120$

$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$

$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$

$g_7(x) = 0,5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 \leq 30$

$g8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$

and $-10 \leq x_i \leq 10(i = 1, \ldots, 10)$.

Optimal Solution: x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927), and $f(x^*) = 24.3062091$.

g08: Maximizing: $f(x) = \frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1+x_2)}$

constraints:

$g_1(x) = x_1^2 - x_2 + 1 \leq 0$

$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$

and $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$.

Optimal Solution: x^* = (1.2279713, 4.2453733), and $f(x^*)$ = 0.095825.

g09: minimizing $f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2$
$+x_7^4 - 4x_6x_7 + 10x_6 - 8x_7$

constraints:

$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$

$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$

$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$

$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$

and $-10 \leq x_i \leq 10 (i = 1, \ldots, 7)$.

Optimal Solution: $x^* =$ (2.330499, 1.951372,-0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227) and $f(x^*) =$ 680.6300573.

g10: minimizing $f(x) = x_1 + x_2 + x_3$

constraints:

$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$

$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$

$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$

$g_4(x) = -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0$

$g_5(x) = -x_2 x_7 + 1250 x_5 + x_1 x_4 - 1250 x_4 \leq 0$

$g_6(x) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0$

and $100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000, (i = 2, 3), 10 \leq x_i \leq 1000, (i = 4, \ldots, 8)$.

Optimal solution: $x^* =$ (579.3167,1359.943, 5110.071,182.0174, 295.5985, 217.9799,286.4162, 395.5979) and $f(x^*) =$ 7049.3307.

Conclusions and Future Remarks

This paper has reviewed some natural computation strategies as a survey concerning optimization strategies. The so-called collaborative model PSO has been exposed in order to understand the ability to extract some biological concepts and apply them in computational models as described along the paper. Such biological inspired models have proof to be a powerful tool in order to solve non common problems in a collaborative/competitive way.

Particle Swarm Optimization is often failed in searching the global optimal solution in the case of the objective function has a large number of dimensions. The reason of this phenomenon is not only existence of the local optimal solutions, the velocities of the particles sometimes lapsed into the degeneracy, so that the successive range is restricted in the sub-plain of the whole search hyper-plain [17]. The sub-plane that is defined by finite number of particle velocities is a partial space in the whole search space. The issue of local optima in PSO has been studied and proposed several modifications on the basic particle driven equation [18; 19]. There used a kind of adaptation technique or randomized method (e.g. mutation in evolutionary computations) to keep particles velocities or to accelerate them. Although such improvements work well and have ability to avoid fall in the local optima, the problem of early convergence by the degeneracy of some dimensions is still remaining, even if there are no local optima. Hence the PSO algorithm does not always work well for the high-dimensional function.

Usually, neural network parameters are in a high-dimensional space and then PSO algorithms are not very efficient ones dealing with such individuals. Best solution could be the integration of PSO and GA in a new model $GPSO$ taking advantages of both models, or at least to improve the impact of the high dimensional individuals in the PSO algorithm, see figures 3 and 4.

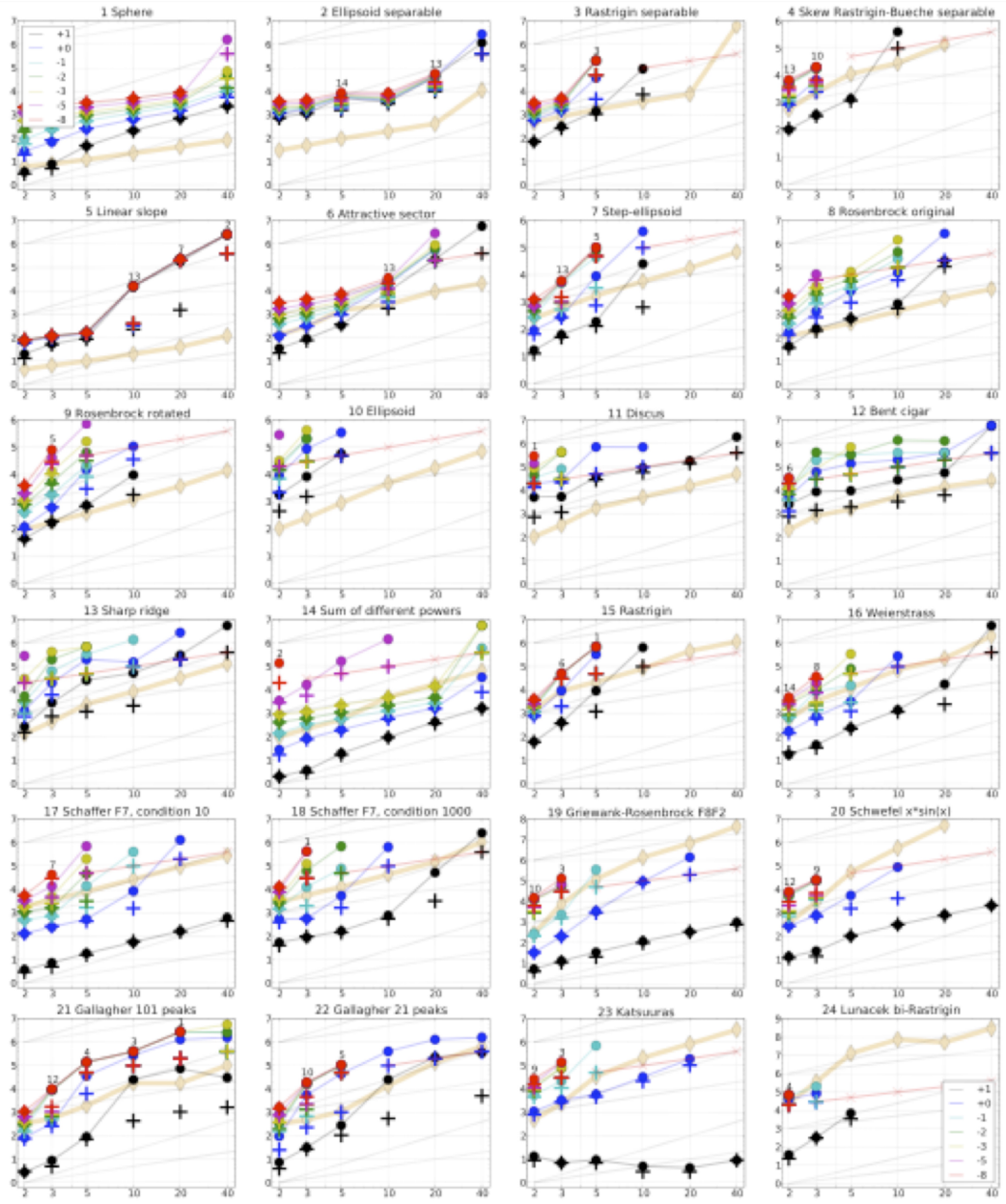

Figure 3: PSO Expected Running Time (ERT, ●) to reach $f_{\mathrm{opt}} + \Delta f$ and median number of f-evaluations from successful trials (+), for $\Delta f = 10^{\{+1,0,-1,-2,-3,-5,-8\}}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. For each function and dimension, $\mathrm{ERT}(\Delta f)$ equals to $\#\mathrm{FEs}(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{\mathrm{opt}} + \Delta f$ was surpassed. The $\#\mathrm{FEs}(\Delta f)$ are the total number (sum) of f-evaluations while $f_{\mathrm{opt}} + \Delta f$ was not surpassed in the trial, from all (successful and unsuccessful) trials, and f_{opt} is the optimal function value. Crosses ($\times$) indicate the total number of f-evaluations, $\#\mathrm{FEs}(-\infty)$, divided by the number of trials. Numbers above ERT-symbols indicate the number of successful trials. Y-axis annotations are decimal logarithms. The thick light line with diamonds shows the single best results from BBOB-2009 for $\Delta f = 10^{-8}$. Additional grid lines show linear and quadratic scaling.

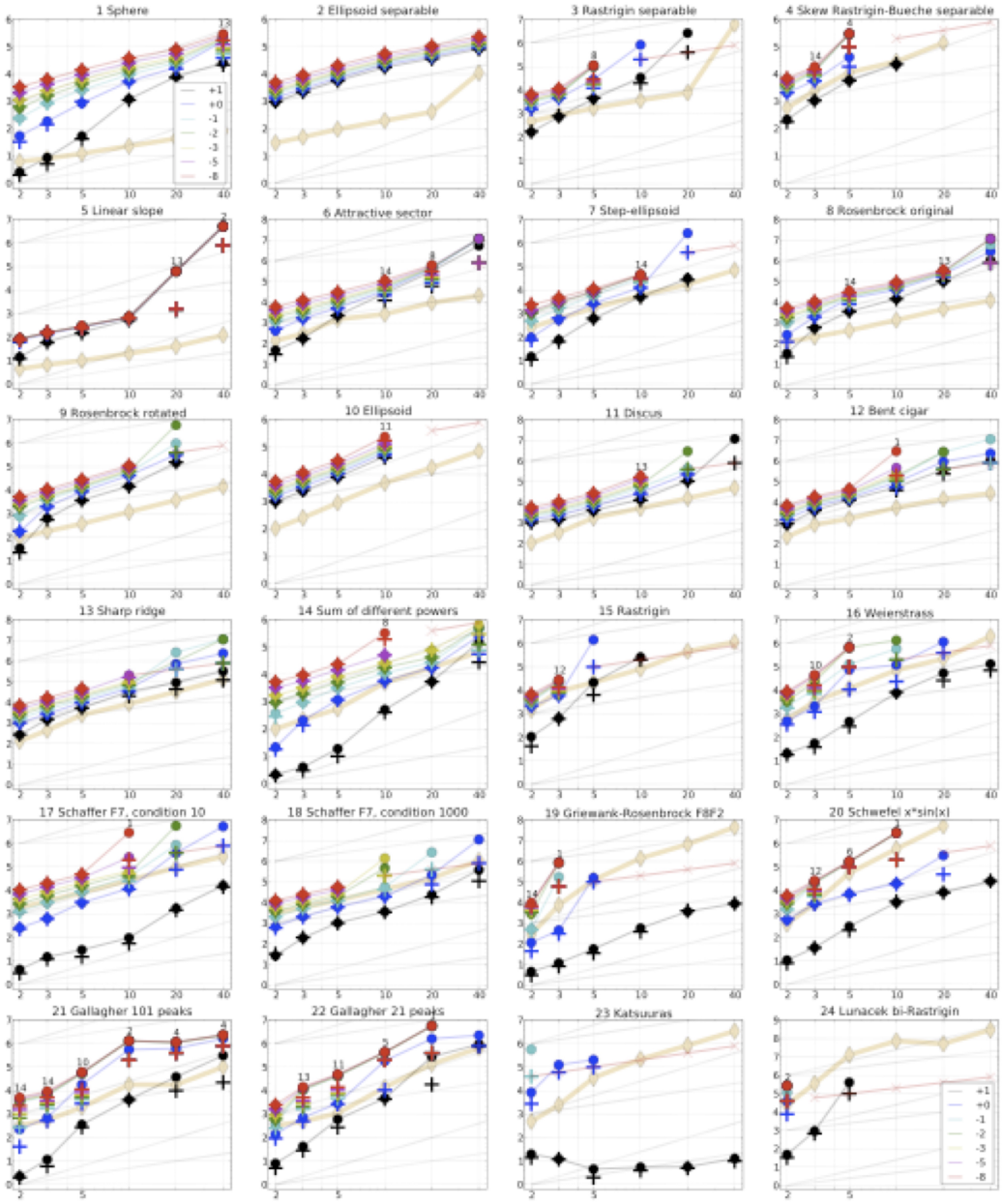

Figure 4: DE-PSO Expected Running Time (ERT, ●) to reach $f_{\mathrm{opt}} + \Delta f$ and median number of f-evaluations from successful trials (+), for $\Delta f = 10^{\{+1,0,-1,-2,-3,-5,-8\}}$ (the exponent is given in the legend of f_1 and f_{24}) versus dimension in log-log presentation. For each function and dimension, $\mathrm{ERT}(\Delta f)$ equals to $\#\mathrm{FEs}(\Delta f)$ divided by the number of successful trials, where a trial is successful if $f_{\mathrm{opt}} + \Delta f$ was surpassed. The $\#\mathrm{FEs}(\Delta f)$ are the total number (sum) of f-evaluations while $f_{\mathrm{opt}} + \Delta f$ was not surpassed in the trial, from all (successful and unsuccessful) trials, and f_{opt} is the optimal function value. Crosses ($\times$) indicate the total number of f-evaluations, $\#\mathrm{FEs}(-\infty)$, divided by the number of trials. Numbers above ERT-symbols indicate the number of successful trials. Y-axis annotations are decimal logarithms. The thick light line with diamonds shows the single best results from BBOB-2009 for $\Delta f = 10^{-8}$. Additional grid lines show linear and quadratic scaling.

Bibliography

[1] Beni G. E. Ahin, Spears W.M., *Swarm Robotics WS 2004, From Swarm Intelligence to Swarm Robotics*. LNCS 3342, pp. 1-9, 2005. Springer-Verlag Berlin Heidelberg, 2005.

[2] Bratton, D., Kennedy, J.,*Defining a Standard for Particle Swarm Optimization*.Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007).

[3] Cagnina, L.C., Esquivel,S. C., Coello, C.,*Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer*, 2008.

[4] Clerc, M., Kennedy, J.,*The particle Swarm: Explosion, Stability, and Convergence in a Multidimensional Complex Space*,IEEE Trans. Evol. Comput., vol. 6, no. 1, pp. 5873, Feb. 2002.

[5] HernÃąndez, A., Muñoz, A., Villa, E., Botello, S., *COPSO: Constrained OptimizaciÃşn Via PSO Algorithm*,ComunicaciÃşn TÃ©cnica No I0704/22022007.(CC/CIMAT), 2007.

[6] Hatanaka, T., Korenaga, T., Kondo, N., Uosaki, K.*Search Performance Improvement for PSO in High Dimensional Space*,2007.

[7] Hvass, P.,M.E.,*Tuning & Simplifying Heuristical Optimization*,Thesis PhD,University of Southampton, 2010.

[8] Hu, X., Eberhart, R., *Solving Constrained Nonlinear Optimization Problems wirh Particle Swarm Optimization*. Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informaticsics, SCI2002 , Vol. 5, IIIS, July 2002.

[9] Kennedy, J., Eberhart, R., *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001.

[10] Michalewicz, Z. , Fogel, D., *How to solve it. Modern Heuristics*. Springer,1998.

[11] Michalewicz, Z. , Shoenauer, M., *Evolutionary Algorithms for Constrained Parameter Optimization Problems*. Evolutionary Computation, 1996, Vol. 4, pp. 132.

[12] Parsopoulos, K.E., Vrahatis, M. N.,Particle Swarm Optimization Method for Constrained Optimization Problems,2002.

[13] Shi, Y., Eberhart, R.,*A Modified Particle Swarm Optimizer*, Proc. IEEE World Congr. Comput. Intell., 1998, pp. 69 73.

[14] Tan, C. , Tapabrata, R. , Pawan, D.,*Supplementary Material on Parameter Estimation using Swarm Algorithm*, Preprint Elsevier Science, 2004.

[15] Weise, T.,*Global Optimization Algorithms-Theory and Application*, e-book, 2009.

[16] Zhan, Z., Zhang, J., Li, Y., Chung, H. S. -H., *Adaptative Particle Swarm Optimization*. IEEE Transactions On Systems, Man, and Cybernetics,Part B: Cybernetics,Digital Object Identifier: 10.1109/TSMCB.2009.2015956, 2009.

[17] Rapaic, M. R., Kanovic, Z. and Jelicic, Z. D., "A Theoretical and Empirical Analysis of Convergence Related Particle Swarm Optimization", *WSEAS Transactions on Systems and Control, 4*, 2009.

[18] Parsopoulos, K., Plagianakos, V. P. and Magoulas, G. D., "Stretching technique for obtaining global minimizers through particle swarm optimization" in *Proceedings of the Particle Swarm Optimization Workshop* (Indianapolis), pp. 22-29, 2001.

[19] Hendtlass, T., "A particle swarm algorithm for high dimensional, multi-optima problem spaces", in *Proceedings of Swarm Intelligence Symposium 2005* (Pasadena), pp. 149-154, 2005.

Authors' Information

Nuria Gómez Blas *- Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing sn, 28031 Madrid, Spain; e-mail: nuria.gomez.blas@upm.es*
Major Fields of Scientific Research: Bio-inspired Algorithms, Natural Computing

Luis Fernando de Mingo López *- Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing sn, 28031 Madrid, Spain; e-mail: fernando.demingo@upm.es*
Major Fields of Scientific Research: Artificial Intelligence, Social Intelligence

Juan Castellanos Peñuela *- Departamento de Inteligencia Artificial, Escuela Técnica Superior de Ingeniería Informática, Universidad Politécnica de Madrid, Campus de Montegancedo sn, 28660 Madrid,, Spain; e-mail: jcastellanos@fi.upm.es*
Major Fields of Scientific Research: Formal Languages, Neural Networks