PARTICLE SWARM OPTIMIZER IN NOISY AND CONTINUOUSLY CHANGING ENVIRONMENTS

K.E. PARSOPOULOS
Department of Mathematics
UPAIRC, University of Patras
GR–26110 Patras, Greece
email: kostasp@math.upatras.gr

M.N. VRAHATIS
Department of Mathematics
UPAIRC, University of Patras
GR–26110 Patras, Greece
email: vrahatis@math.upatras.gr

ABSTRACT
In this paper we study the performance of the recently proposed Particle Swarm optimization method in the presence of noisy and continuously changing environments. Experimental results for well known and widely used optimization test functions are given and discussed. Conclusions for its ability to cope with such environments as well as real–life applications are also derived.

KEY WORDS
Particle Swarm, Evolutionary Methods, Noisy Functions

1. INTRODUCTION

Optimization techniques are of undisputed importance in science and technology. They can be used for many purposes: optimal design of systems, optimal operation of systems, determination of performance limitations of systems, or simply the solution of sets of equations. Many recent advances in science, economics and engineering rely on numerical techniques for computing globally optimal solutions to corresponding optimization problems [8]. Due to the existence of multiple local and global optima all these problems cannot be solved by classical nonlinear programming techniques.

During the past three decades, however, many new algorithms have been evolved and new approaches have been implemented, resulting to powerful optimization algorithms such as the Evolutionary Algorithms [11]. Differently from other adaptive algorithms, evolutionary techniques work on a set of potential solutions, which is called *population*, and find the optimal solution through cooperation and competition among the potential solutions. These techniques can often find optima in complicated optimization problems more quickly than traditional optimization methods. The most commonly used population–based evolutionary computation techniques, such as Genetic Algorithms and Artificial Life methods, are motivated from the evolution of nature and the social behavior of humans and insects.

It is worth noticing that, in general, Global Optimization (GO) strategies possess strong theoretical convergence properties, and, at least in principle, are straightforward to implement and apply. Issues related to their numerical efficiency are considered by equipping GO algorithms with a "traditional" local optimization phase. Global convergence, however, needs to be guaranteed by the global–scope algorithm component which, theoretically, should be used in a complete, "exhaustive" fashion. These remarks indicate the inherent computational demand of the GO algorithms, which increases non–polynomially, as a function of problem–size, even in the simplest cases.

In practical applications, most of the above–mentioned methods can detect just *sub–optimal solutions* of the objective function. In many cases these sub–optimal solutions are acceptable but there are applications where an optimal solution is not only desirable but also indispensable. Moreover, in many applications there are imprecise values for the input data as well as for the function values. Therefore, the development of robust and efficient GO methods for dynamic environments such as the aforementioned, is a subject of considerable ongoing research [22].

Recently, Eberhart and Kennedy (1995) proposed the *Particle Swarm Optimization* (PSO) algorithm [9]: a new, simple evolutionary algorithm, which differs from other evolution–motivated evolutionary computation techniques in that it is motivated from the simulation of birds' social behavior. Although, in general, PSO results in global solutions even in high–dimensional and multimodal spaces [15, 16, 17], there are not many results about its behavior in the presence of noise, i.e. the performance of the method when noise is inserted into the function values and/or the landscape is continuously changing.

The remaining of the paper is organized as follows: in Section 2 we give a discussion of optimization of noisy functions as well as a simulation of the influence of noise (proportional to a Gaussian distributed random number with zero mean and various variances). In Section 3 a brief overview of the PSO method is presented, while in Section 4 numerical results are presented. Finally, in Section 5, we give some concluding remarks.

2. OPTIMIZATION OF NOISY FUNCTIONS

Several methods for finding the extrema of a function $f: \overline{\mathcal{D}} \subset R^n \rightarrow R$, where $\mathcal{D}$ is open and bounded, have been proposed, with many applications in different scientific fields (mathematics, physics, engineering, computer science etc.). Most of them require precise function and gradient values. In many applications though, precise values are either impossible or time consuming to obtain. For example, when the function and gradient values depend on the results of numerical simulations, then it may be difficult or impossible to get very precise values. Or, in other cases, it may be necessary to integrate numerically a system of differential equations in order to obtain a function value, so that the precision of the computed value is limited. Furthermore, in many problems the accurate values of the function to be minimized are computationally expensive. Such problems are common in real life applications as in the optimization of parameters in chemical experiments or finite element calculations. With such applications in mind, robust methods are needed, which make good progress with the fewest possible number of function evaluations.

The problem of optimization of noisy or imprecise (not exactly known) functions occurs in various applications, as, for instance, in the task of experimental optimization. Also, the problem of locating local maxima and minima of a function from approximate measurement results is vital for many physical applications. In spectral analysis, chemical species are identified by locating local maxima of the spectra. In radioastronomy, sources of celestial radio emission and their subcomponents are identified by locating local maxima of the measured brightness of the radio sky. Elementary particles are identified by locating local maxima of the experimental curves.

The theory of local optimization provides a large variety of efficient and effective methods for the computation of an optimizer of a smooth function f. For example, Newton–type and quasi–Newton methods show superlinear convergence in the vicinity of a nondegenerate optimizer. However, these methods require the Hessian or the gradient, respectively, in contrast to other optimization procedures, like the simplex method [13], the direction set method of Powell [7], or some other recently proposed methods [5, 6, 23].

In some applications, however, the function to be minimized is only known within some (often unknown and low) precision. This might be due to the fact that evaluation of the function means measuring some physical or chemical quantity or performing a finite element calculation in order to solve partial differential equations. The function values obtained are corrupted by noise, namely stochastic measurement errors or discretization errors. This means that, although the underlying function is smooth, the function values available show a discontinuous behavior. Moreover, no gradient information is available. For small variances in a neighborhood of a point the corresponding function values reflect the local behavior of the noise rather than that of the function. Thus, a finite–difference procedure to estimate the gradient fails [5].

The traditional method for optimizing noisy functions is the simplex or polytope method by Nelder and Mead [7, 13, 14, 19]. This method surpasses other well–known optimization methods when dealing with the large noise case. However, this is not valid in the noiseless case. The ability of this method to cope with noise is due to the fact that it proceeds solely by comparing the relative size of the function values, as the proposed method does. The Simplex method does not use a local model of the function f and works without the assumption of continuity. Although this method has poor convergence properties (for a convergence proof of a modified version see [21]), it has been a useful method in many sequential applications, but it is difficult and inefficient to implement in parallel. The method can be deficient when the current simplex is very "flat". This can be avoided by suitable variants (see for example [21]). More sophisticated methods in this direction are discussed by Powell [20].

To study the influence of the imprecise information (regarding the values of the objective function), we simulate imprecisions with the following approach. Information about $f(x)$ is obtained in the form of $f^{\sigma}(x)$, where $f^{\sigma}(x)$ is an approximation to the true function value $f(x)$, contaminated by a small amount of noise η. Specifically, the function values are obtained as [6]:

$$f^{\sigma}(x) = f(x)(1+\eta), \qquad \eta \sim N(0,\sigma^2), \qquad (1)$$

where $\eta \sim N(0,\sigma^2)$ denotes a Gaussian distributed random variable with zero mean and variance σ^2, i.e., relative stochastic errors are used for the test problems. To obtain η, we apply the method of Box and Muller [2], using various values of the variance σ^2.

3. THE PARTICLE SWARM OPTIMIZER

As already mentioned, PSO is different from other evolutionary algorithms. Indeed, in PSO the population dynamics simulates a "bird flock's" behavior where social sharing of information takes place and individuals can profit from the discoveries and previous experience of all other companions during the search for food. Thus, each companion, called *particle*, in the population, which is now called *swarm*, is assumed to "fly" over the search space in order to find promising regions of the landscape. For example, in the minimization case, such regions possess lower function values than other visited previously. In this context, each particle is treated as a point in a D–dimensional space which adjusts its own "flying" according to its flying experience as well as the flying experience of other particles (companions).

There are many variants of the PSO proposed so far, after Eberhart and Kennedy introduced this technique [4, 9]. In our experiments we used a new version of this algorithm, which is derived by adding a new inertia weight to

the original PSO dynamics [3]. This version is described in the following paragraphs.

First let us define the notation adopted in this paper: the i-th particle of the swarm is represented by the D–dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and the best particle in the swarm, i.e. the particle with the smallest function value, is denoted by the index g. The best previous position (the position giving the best function value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the position change (velocity) of the i-th particle is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

The particles are manipulated according to the equations

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}), \quad (2)$$

$$x_{id} = x_{id} + v_{id}, \quad (3)$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$ and N is the size of the population; w is the inertia weight; c_1 and c_2 are two positive constants; r_1 and r_2 are two random values into the range $[0, 1]$.

The first equation is used to calculate i-th particle's new velocity by taking into consideration three terms: the particle's previous velocity, the distance between the particle's best previous and current position, and, finally, the distance between swarm's best experience (the position of the best particle in the swarm) and i-th particle's current position. Then, following the second equation, the i-th particle flies toward a new position. In general, the performance of each particle is measured according to a predefined fitness function, which is problem–dependent.

The role of the inertia weight w is considered very important in PSO convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. In this way, the parameter w regulates the trade–off between the global (wide–ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e. fine–tuning the current search area. A suitable value for the inertia weight w usually provides balance between global and local exploration abilities and consequently a reduction on the number of iterations required to locate the optimum solution. A general rule of thumb suggests that it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually decrease it to get more refined solutions, thus a time decreasing inertia weight value is used. The initial population can be generated either randomly or by using a Sobol sequence generator which ensures that the D-dimensional vectors will be uniformly distributed into the search space [18].

From the above discussion it is obvious that PSO, to some extent, resembles evolutionary programming. However, in PSO, instead of using genetic operators, each individual (particle) updates its own position based on its own search experience and other individuals (companions) experience and discoveries. Adding the velocity term to the current position, in order to generate the next position, resembles the mutation operation in evolutionary programming. Note that in PSO, however, the "mutation" operator is guided by the particle's own "flying" experience and benefits by the swarm's "flying" experience. In another words, PSO is considered as performing mutation with a "conscience", as pointed out by Eberhart and Shi [3].

4. EXPERIMENTAL RESULTS OF THE PSO METHOD IN CHANGING SEARCH SPACES

We consider three simple and well-known optimization test problems [12] to check the performance of the PSO method. During the evolution of the swarm we add noise in terms of multiplying the population by a rotation matrix (and thus rotating the whole search space including the global minimizer) and adding a Gaussian distributed random term to the function values, according to Relation (1). The rotation angle is taken randomly between 0 and 360 degrees, and the standard deviation σ of the random term added to the function values increases from 0 to 0.9. For each test function, a population of size 20 and default values for the parameters c_1 and c_2 of PSO have been used: $c_1 = c_2 = 0.5$. There have been done 100 runs for each different value of the noise's standard deviation. A time decreasing inertia weight value, i.e. starting from 1 and gradually decreasing towards 0.4, has been found to work better than using a constant value. This is because large inertia weights help to find good seeds at the beginning of the search, while, later, small inertia weights facilitate a finer search. The desired accuracy for finding the global minimum has been 10^{-3}.

The first test function considered is the *Rosenbrock* function which is defined by the formula [12]:

$$f(x_1, x_2) = (10^4 x_1 x_2 - 1)^2 + (1 - x_1)^2, \quad (4)$$

and has the global minimizer $x^* = (1, 1)$ with function value $f(x^*) = 0$. The initial population has been taken into the interval $[0, 2]^2$ for each run and the results are given in Table 1. Each row of the Table contains the success rate and the mean number of PSO cycles needed to detect

Noise's St. Dev. σ	Success Rate	Mean Cycles
0	100%	406.90
0.1	76%	2222.84
0.2	88%	2159.95
0.3	88%	2064.81
0.4	76%	1623.21
0.5	60%	2136.86
0.7	40%	2030.40
0.9	76%	1684.57

Table 1. Analysis of the results for the minimization of the Rosenbrock function.

the global minimizer of the function for the corresponding values of noise standard deviation σ. The zero value of standard deviation in the first row of Table 1, denotes the plain PSO performance (without noise addition and rotation). The rest columns refer to results obtained by adding noise with standard deviation values between 0.1 and 0.9, while simultaneously rotating the search space by a random angle. It is clear that increasing the noise standard deviation causes no serious instability to the method and does not decrease significantly the success rate, except for some values of the variance between 0.5 and 0.7, where a remarkable decrease at the success rate is observed. Failures denote that PSO has not been able to find the global minimizer into the maximum number of iterations (cycles of the method) which has been set to 5000. From these results one suspects that PSO is a well noise tolerant method.

The same good behaviour is observed for the *Levy No.5* function, which is given by the formula [10]:

$$f(x) = \sum_{i=1}^{5} i\cos[(i+1)x_1+i] \times \sum_{j=1}^{5} j\cos[(j+1)x_2+j] + \\ +(x_1+1.42513)^2+(x_2+0.80032)^2, \quad (5)$$

where $-10 \leq x_i \leq 10, i = 1, 2$. There are about 760 local minimizers and one global minimizer $x^* = (-1.3068, -1.4248)$ with function value $f(x^*) = -176.1375$. The large number of local optimizers makes extremely difficult for various methods to locate the global minimizer. The results for this function are given in Table 2. The initial population has been taken into the interval $[-2, 2]^2$ for each run. As can be seen, addition of noise even of large standard deviation does not affect the success rate of the method significantly and in some cases it helps avoiding local minima of the objective function.

Noise's St. Dev. σ	Success Rate	Mean Cycles
0	100%	624.03
0.1	100%	992.00
0.2	100%	1393.60
0.3	96%	894.79
0.4	100%	984.64
0.5	96%	1255.83
0.7	100%	1204.20
0.9	96%	1498.33

Table 2. Analysis of the results for the minimization of the Levy No.5 function.

Proportional results are obtained in the third experiment. The test function considered here is the *Beale* function [12]:

$$f(x_1,x_2) = [y_1 - x_1(1-x_2)]^2 + [y_2 - x_1(1-x_2^2)]^2 + \\ +[y_3 - x_1(1-x_2^3)]^2, \quad (6)$$

where $y_1 = 1.5, y_2 = 2.25$ and $y_3 = 2.625$. This function has a global minimizer $x^* = (3, 0.5)$ with function value $f(x^*) = 0$. The results for this function are exhibited in Table 3.

Noise's St. Dev. σ	Success Rate	Mean Cycles
0	90%	1807.92
0.1	96%	1963.29
0.2	96%	1543.04
0.3	88%	2125.31
0.4	88%	2153.00
0.5	68%	1960.88
0.7	56%	2453.00
0.9	80%	2037.80

Table 3. Analysis of the results for the minimization of the Beale function.

If we decrease, now, the desired accuracy to 10^{-6}, we observe that the swarm moves closely to the global minimizer of each test function but it cannot find it with the desired accuracy. This is more clear if we add an offset to the original global minimizer's position, at each iteration, as performed by Angeline [1]. The mean function values of the swarm for each iteration, after 100 runs, for the aforementioned test problems, are exhibited in Figs. 1, 2 and 3. Different line styles in the figures correspond to different values of the offset. For all runs, a value of the noise variance equal to 0.01 was used. The vertical axis of each plot is logarithmically scaled to facilitate visual comparison. In Figs. 1 and 3 the logarithm $\log_{10}$ of the swarm's mean function value for 100 runs is exhibited, while in Fig. 2 the logarithm $\log_{10}$ of the swarm's mean absolute error is exhibited, due to the negative values of the Levy No.5 function.

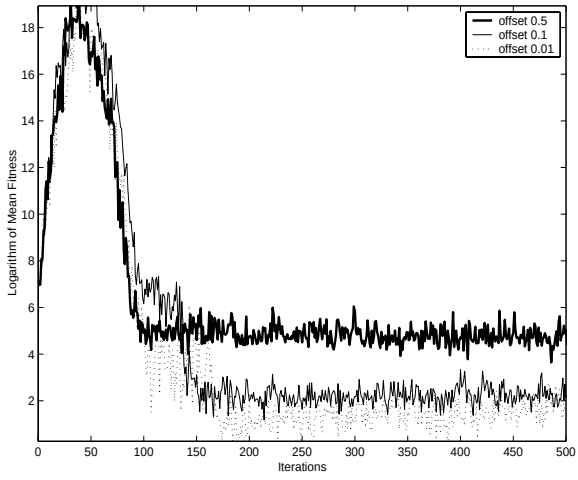

Figure 1. Mean fitness value for the Rosenbrock function.

It is clear now that noise addition causes no crucial instability to the PSO algorithm. Even in very scattered landscapes and multimodal functions the results are very promising. Thus, it would be very interesting to check the

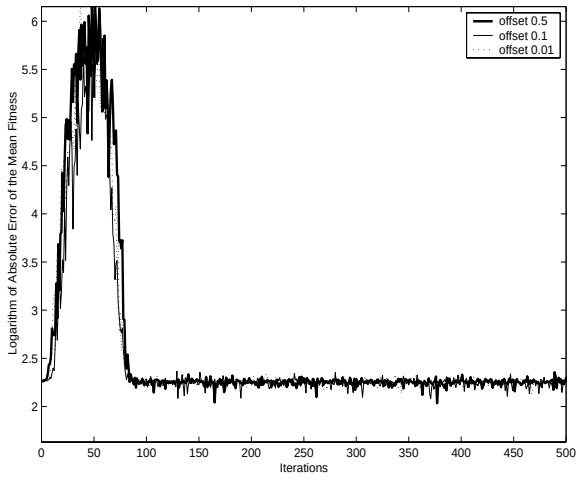

Figure 2. Mean fitness value for the Levy No.5 function.

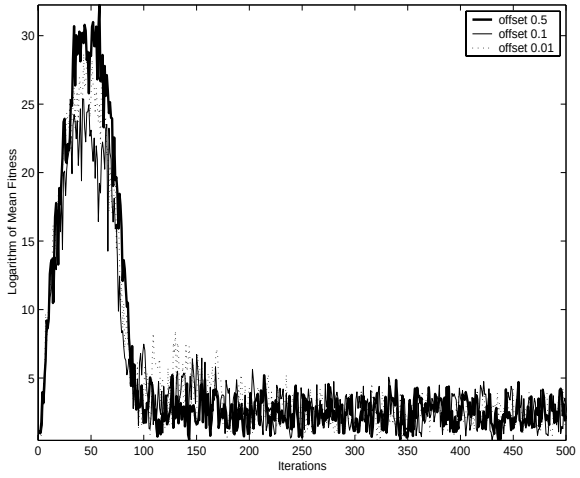

Figure 3. Mean fitness value for the Beale function.

performance of PSO in real–life applications where noise to the input data and to the function values is almost always present.

5. CONCLUSIONS

A study of the ability of the Particle Swarm optimization method to cope with continuously changing environments has been given. The experimental results indicate that in the presence of noise and when the landscape is continuously changing PSO method is very stable and efficient. In fact, in many cases, the presence of noise seems to help PSO to avoid local minima of the objective function and locate the global one. Even in the cases where the standard deviation of the noise was large and a fixed offset was added to the global minimizer's position at each iteration, PSO was able to move closely to the global minimizer's position. Thus, preliminary results indicate that PSO has the ability to cope with noisy and continuously changing environments. Further work shall be done to check the performance of PSO in other dynamic environments and real–life applications.

References

[1] P. Angeline, Tracking extrema in dynamic environments, *Proc. Int. Conf. Evol. Progr.*, Indianapolis, Indiana, USA, 1997.

[2] G.E.P. Box, M.E. Muller, A note on the generation of random normal deviates, *Ann. Math. Statistics*, 29, 1958, 610–611.

[3] R.C. Eberhart, Y.H. Shi, Evolving Artificial Neural Networks, *Proc. Int. Conf. Neural Networks and Brain*, Beijing, P.R. China, 1998, PL5-PL13.

[4] R.C. Eberhart, P.K. Simpson, R.W. Dobbins, *Computational Intelligence PC Tools* (Boston: Academic Press Professional, 1996).

[5] C. Elster, A. Neumaier, A grid algorithm for bound constrained optimization of noisy functions, *IMA J. Numer. Anal.*, 15, 1995, 585–608.

[6] C. Elster, A. Neumaier, A method of trust region type for minimizing noisy functions, *Computing*, 58, 1997, 31–46.

[7] R. Fletcher, *Practical Methods of Optimization* (New York: John Wiley, 1987).

[8] R. Horst, P.M. Pardalos, N.V. Thoai, *Introduction to Global Optimization* (Kluwer Academic Publishers, 1995).

[9] J. Kennedy, R.C. Eberhart, Particle Swarm Optimization, *Proc. IEEE Int. Conf. Neural Networks*, Piscataway, NJ, USA, 1995, 1942–1948.

[10] A. Levy, A. Montalvo, S. Gomez, A. Galderon, *Topics in Global Optimization* (Lecture Notes in Mathematics No. 909., New York: Springer–Verlag, 1981).

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (New York: Springer, 1996).

[12] J.J. More, B.S. Garbow, K.E. Hillstrom, Testing Unconstrained Optimization Software, *ACM Trans. Math. Soft.*, 7(1), 1981, 17–41.

[13] J.A. Nelder, R. Mead, A simplex method for function minimization, *Computer J.*, 7, 1965, 308–313.

[14] J. Nocedal, Theory of algorithms for unconstrained optimization, in A. Iserles (Ed.), *Acta Numerica* (Cambridge: Cambridge University Press, 1992), 199–242.

[15] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Objective function "stretching" to alleviate convergence to local minima, *Nonlinear Analysis, TMA*, to appear in 2001.

[16] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Stretching technique for obtaining global minimizers through Particle Swarm Optimization, *Proc. of the Particle Swarm Optimization Workshop*, Indianapolis, Indiana, USA, 2001, 22-29.

[17] K.E. Parsopoulos, M.N. Vrahatis, Modification of the Particle Swarm Optimizer for locating all the global minima, *Proc. Int. Conf. Artificial Neural Networks and Genetic Algorithms (ICANNGA)*, Prague, Czech Republic, to appear in 2001 by Springer.

[18] W.H. Press, W.T. Vetterling, S.A. Teukolsky, B.P. Flannery, *Numerical Recipes in Fortran 77* (Cambridge: Cambridge University Press, 1992).

[19] M.J.D. Powell, A review of algorithms for nonlinear equations and unconstrained optimization, *Proc. ICIAM*, Philadelphia, USA, 1987, 220–232.

[20] M.J.D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, *Numerical Analysis Reports*, DAMTP 1992/NA5, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 1992.

[21] V. Torczon, On the convergence of the multidimensional search algorithm, *SIAM J. Optimization*, 1, 1991, 123–145.

[22] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos, G.D. Magoulas, A class of gradient unconstrained minimization algorithms with adaptive stepsize, *J. Comp. Appl. Math.*, 114, 2000, 367–386.

[23] M.N. Vrahatis, G.S. Androulakis, M.E. Manoussakis, A new unconstrained optimization method for imprecise function and gradient values, *J. Math. Anal. Appl.*, 197, 1996, 586–607.