Initializing the Particle Swarm Optimizer Using the Nonlinear Simplex Method

K.E. PARSOPOULOS, M.N. VRAHATIS
Department of Mathematics
University of Patras
University of Patras Artificial Intelligence Research Center (UPAIRC)
GR-261.10 Patras
GREECE

Abstract: - Initialization of the population in Evolutionary Computation algorithms is an issue of ongoing research. Proper initialization may help the algorithm to explore the search space more efficiently and detect better solutions. In this paper, the Nonlinear Simplex Method is used to initialize the swarm of the Particle Swarm technique. Experiments for several well-known benchmark problems imply that better convergence rates and success rates can be achieved by initializing the swarm this way.

Key-Words: - Particle Swarm, Nonlinear Simplex Method, Optimization

1 Introduction

Evolutionary Computation (EC) algorithms provide solutions to many hard optimization problems that are very difficult to cope with using the traditional Gradient based methods, due to their nature that may imply discontinuities of the search space, non-differentiable objective functions, imprecise arguments and function values. The main advantage of these algorithms is the usage of a population of potential solutions that explore the search space simultaneously, exchanging information among them and using only function values and not derivatives of the objective function.

The most well-known paradigm of EC algorithms are the Genetic Algorithms (GA) that are used widely, especially in engineering and industrial applications [1, 2, 3]. According to the GA's theory, the population is binary encoded and genetic operators, inspired by the human DNA evolution procedures, are applied to the population in order to evolve it and thus explore the search space efficiently. Several other ideas (Evolutionary Programming [4], Evolution Strategies [5, 6], Genetic Programming [7]) are inspired by the GAs and they are widely applied exhibiting significant results in several scientific fields.

Recently, a new research field arised, called Swarm Intelligence (SI). SI argues that intelligent human cognition derives from the interaction of individuals in a social environment and that the main ideas of sociocognition can be effectively applied to develop stable and efficient algorithms for optimization tasks [8]. Ant Colony Optimization (ACO) is the most well known SI algorithm and is mainly used for Combinatorial Optimization tasks, exhibiting very interesting results in experiments as well as in real life applications [9, 10].

The Particle Swarm Optimization (PSO) technique is an SI technique, which is mainly used for Continuous Optimization tasks and has been originally developed by R.C. Eberhart and J. Kennedy [8, 11]. In this technique, the population of potential solutions is called *swarm* and it explores the search space simulating the movement of a "birds' flock" while searching for food, where global exchange of information among all individuals, which are called *particles*, takes place and each particle can profit from the discoveries of the rest of the swarm. PSO has been proved to be very efficient algorithm in solving hard optimization problems and engineering applications, including neural networks training and Human Tremor analysis. Many variants and techniques have been developed to improve further its performance [8, 12, 13, 14, 15, 16].

The initialization of the swarm is usually done using a uniform distribution over the search space, but this is not always the best way. In another approach a Sobol sequence generator is used to generate the initial swarm in order to be uniformly distributed over multidimensional search spaces [14]. Proper initialization of the swarm seems to help PSO to explore efficiently the search space and detect solutions of better quality.

In this paper, the Nonlinear Simplex Method developed by J.A. Nelder and R. Mead [17] is used to initialize the swarm of the PSO technique. Experimental results for many well-known test functions imply that this is a very promising way of initialization and that it can significantly improve the convergence rates and in some cases the success rate of the PSO. In the next section, the Nonlinear Simplex Method is described. In Section 3 the PSO technique is briefly analyzed and in Section 4 the proposed algorithm is described and applied in several test functions. The paper closes with some conclusions and ideas for further work in Section 5.

2 The Nonlinear Simplex Method

The Nonlinear Simplex Method (NSM) was developed by J.A. Nelder and R. Mead [17] for function minimization tasks. It needs only function evaluations and there is no need for derivatives computation. In general the NSM is considered as the best method if the figure of merit is "get something to work quickly", especially when noisy problems are considered, and has a geometrical naturalness, which makes it delightful to work through [18].

A D-dimensional *simplex* is a geometrical figure consisting of D+1 vertices (D-dimensional points) and all their interconnecting segments, polygonal faces etc. Thus, in two dimensions, a simplex is a triangle, while in three dimensions it is a tetrahedron. In general, we consider only simplexes that are *non-degenerated*, i.e., that enclose a finite inner D-dimensional volume. If any vertex of such a simplex is taken as the origin, then the rest D vertices define vector directions that span the D-dimensional vector space.

The NSM starts with an initial simplex and takes a series of steps where the vertex of the simplex with the highest function value is mostly moved through the opposite face of the simplex to a lower point. If this is possible, then the simplex is expanded in one or another direction to take larger steps. When the method reaches a "valley floor", the simplex is contracted in the transverse direction in order to ooze down the valley or it can be contracted in all directions, pulling itself in around its lowest point. A subroutine implementing NSM in Fortran 77 is given in [18]. The possible moves of a simplex are shown in Fig.1.

When applied to function minimization problems, the NSM is usually restarted at a minimizer that has already been found by the technique, in order to make sure that the stopping criteria have not been fooled by a single anomalous step. Thus, an initial simplex is generated using the found minimizer as one of its vertices and generating the rest D points randomly. Then the algorithm is applied as usual to that simplex. The restart is not expected computationally expensive, since the algorithm had already converged to one of the initial simplex vertices before restarted.

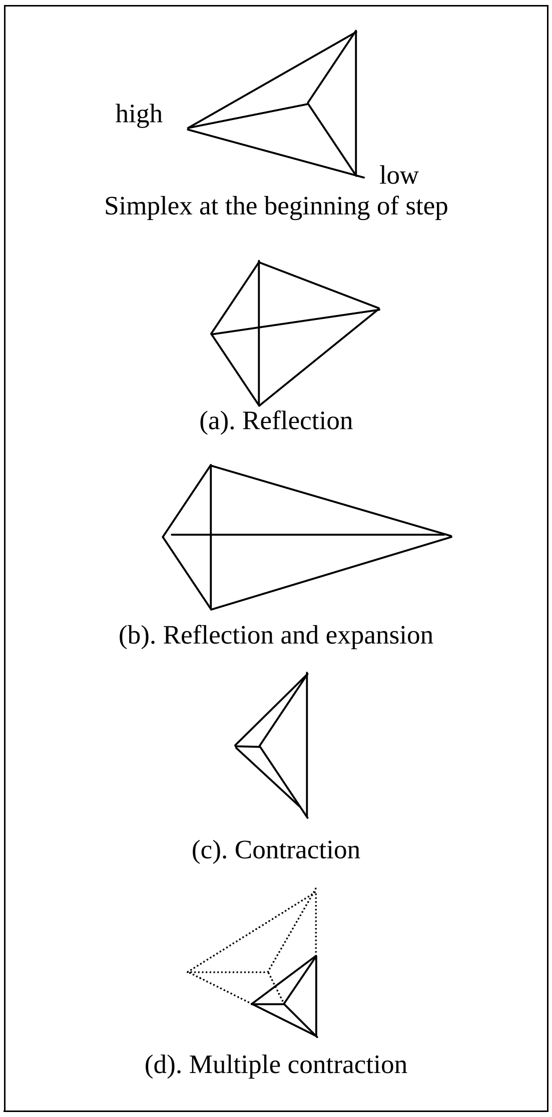
high
low
Simplex at the beginning of step
(a). Reflection
(b). Reflection and expansion
(c). Contraction
(d). Multiple contraction

Fig.1. Possible outcomes for a step in the NSM. The simplex initially is a tetrahedron (top). At the next step it can be (a) a reflection away from the high point, (b) a reflection and expansion away from the high point, (c) a contraction along one dimension from the high point, or (d) a contraction along all dimensions toward the lowest point.

The convergence properties of the NSM are in general poor (for a convergence proof of a modified version see [19]) but in many applications it has been a very useful method, especially in cases of noisy functions and problems with imprecise data. A more efficient variant of the NSM can be found in

[19]. In the next section, the PSO technique is described and discussed.

3 The Particle Swarm Optimization Technique

As already mentioned in the Introduction, that PSO is an Evolutionary Technique (more precisely a Swarm Intelligence technique) but it differs significantly from the GAs. In PSO, there are no DNA inspired operators applied on the swarm. Instead, each particle is assumed to “fly” over the search space in order to find promising regions of the landscape and adjusts its own “flying” according to its “flying” experience as well as the experience of the other particles. The promising regions are characterized by lower function values in the simple function minimization case.

There are many variants of the PSO technique developed so far. In our experiments a version of the algorithm derived by adding an inertia weight to the original PSO dynamics [20] has been used and this version is described in the following paragraphs.

Assuming that the search space is D-dimensional, we denote by $X_i = (x_{i1}, x_{i2},..., x_{iD})$ the i^{th} particle of the swarm and by $P_i = (p_{i1}, p_{i2},..., p_{iD})$ the best position it ever had into the search space, i.e., the position with the smallest function value. Let g be the index of the best particle in the swarm, i.e., the particle with the smallest function value, and $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$ the velocity (position change) of the i^{th} particle.

Then, the swarm is manipulated according to the equations

$$v_{id} = w\, v_{id} + c_1\, r_1\, (p_{id} - x_{id}) + c_2\, r_2\, (p_{gd} - x_{id}), \quad (1)$$
$$x_{id} = x_{id} + \div v_{id} \quad (2)$$

where $d=1,2,...,D$; $i=1,2,...,N$ 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], and $\div$ is a constriction factor used to control the magnitude of the velocity (in unconstrained problems it is usually set equal to one).

The first equation is used to calculate the i^{th} particle’s new velocity and it takes into consideration three main terms: the particle’s previous velocity, the distance of the particle’s current position from its own best position, and the distance of the particle’s current position from the swarm’s best experience (position of the best particle). Then, the particle moves to a new position according to the second equation. The performance of each particle is measured using a predefined fitness function, which, in general, is problem dependent.

The inertia weight w plays an important role for the convergence behaviour of the technique. It is used to control the impact of the previous history of velocities to the current velocity of each particle, regulating this way the trade-off between the global and local exploration abilities of the swarm, since large values of w facilitate global exploration of the search space (visiting new regions) while small values facilitate local exploration, i.e., fine-tuning the current search area. Thus, it is better to use large values of w at the first steps of the algorithm and gradually decrease it during the optimization in order to perform more refined search of the already detected promising regions.

From the above discussion it is obvious that PSO resembles, to some extent, the mutation operator of GAs, but since each particle is guided by its own experience and the best experience of the whole swarm, we could say that PSO performs “mutation with conscience”, as pointed out in [20].

The initialization of the swarm is considered as an issue of crucial importance for the PSO’s performance and thus it is an issue of ongoing research. Usually the particles are uniformly distributed over the search space either by using a simple uniform distribution or by using a Sobol sequence generator. In the next section a new way to initialize the swarm is proposed and promising experimental results are exhibited.

4 The Proposed Algorithm and Experimental Results

Providing the initial swarm with some extra information concerning the position of the promising regions into the search space is generally considered as important help for the PSO algorithm, since it may lead to faster convergence and better quality of the solutions provided by the algorithm. The main idea presented in this paper is the usage of the NSM for the generation of the initial swarm.

Suppose that we start with an initial simplex into the D-dimensional search space, the D+1 vertices of the simplex will be the first D+1 particles of the swarm. Then, we apply the NSM Method for N-$(D+1)$ steps, where N is the desired size of the swarm. At each step we assume the new vertex provided by the single NSM step as a new particle and we add it into the swarm. Thus, the initial swarm is provided with the information of the good regions that possess each particle as a vertex of the

NSM simplex in each step. The algorithm is not computationally expensive, since for each particle of the initial swarm one function evaluation is done, which is inevitable even if we use a randomly distributed initial swarm. Furthermore, the experimental results that are presented and discussed in the following paragraphs imply that the convergence rate of the PSO is improved and in some cases the success rate of the algorithm is dramatically increased.

The test functions on which the proposed algorithm has been tested, as well as the dimension of each one and the corresponding reference are given in Table 1.

Function Name	Dimension	Reference
Banana Valley	2	[23]
Branin	2	[3]
Six-hump Camel	2	[3]
DeJong	2	[22]
FreuNSMein-Roth	2	[21]
GolNSMein-Price	2	[3]
Grienwangk	2	[22]
Levy No.3	2	[24]
Levy No.5	2	[24]
Helical Valley	3	[21]
6D Hyper-Ellipsoid	6	[22]
6D Rastrigin	6	[22]
6D Watson	6	[21]
9D Hyper-Ellipsoid	9	[22]

Table 1. The test functions used, their dimensions and corresponding references.

Function Name	Swarm's Size	Initial Interval
Banana Valley	20	[-10,10]
Branin	20	[-5,5]
Six-hump Camel	20	[-10,10]
DeJong	20	[-5,5]
FreuNSMein-Roth	20	[-5,5]
GolNSMein-Price	20	[-10,10]
Grienwangk	20	[-10,10]
Levy No.3	30	[-2,2]
Levy No.5	20	[-5,5]
Helical Valley	30	[-0.5,0.5]
6D Hyper-Ellipsoid	50	[-6,6]
6D Rastrigin	60	[-0.5,0.5]
6D Watson	60	[-1,1]
9D Hyper-Ellipsoid	50	[-5,5]

Table 2. The test functions used, the corresponding swarm's sizes and the initial intervals for the NSM.

For all test functions 100 experiments have been done and the desired accuracy is 10^{-3}. The swarm's size as well as the interval in which the NSM has been initialized are different for each test function and they are given in Table 2.

The maximum number of PSO iterations is 150 for all runs and $c_1 = c_2 = 5$. The inertia weight w was initially set to 1.2 and gradually decreased toward 0.4.

Function Name	Mean Func. Eval.		Impr. (%)
	NS-PSO	PSO	
Banana Valley	1362.4	1466.4	7.1 %
Branin	971.2	1193.6	18.6 %
Six-hump Camel	1064	1154	7.8 %
DeJong	796.4	1019.2	21.9 %
FreuNSMein-Roth	1286.4	1392.4	7.6 %
GolNSMein-Price	1246	1327.6	6.1 %
Grienwangk	704.8	1075.2	34.4 %
Levy No.3	3097.2	3418.8	9.4 %
Levy No.5	2689.2	2820	4.6 %
Helical Valley	1965	2022	2.8 %
6D Hyper-Ellipsoid	3099	3344	7.3 %
6D Rastrigin	5568	6540	14.9 %
6D Watson	5064	5253.6	3.6 %
9D Hyper-Ellipsoid	3704	3924	5.6 %

Table 3. The mean number of function evaluations for 100 runs for each test function and the corresponding improvement percentage. The proposed algorithm is denoted as NS-PSO.

Function Name	Mean Iterations		Impr. (%)
	NS-PSO	PSO	
Banana Valley	67.12	72.32	7.2 %
Branin	47.56	58.68	19 %
Six-hump Camel	52.2	56.7	7.9 %
DeJong	38.82	49.96	22.3 %
FreuNSMein-Roth	63.32	68.62	7.7 %
GolNSMein-Price	61.3	65.38	6.2 %
Grienwangk	34.24	52.76	35.1 %
Levy No.3	102.24	112.96	9.5 %
Levy No.5	88.64	93	4.7 %
Helical Valley	64.5	66.4	2.9 %
6D Hyper-Ellipsoid	60.98	65.88	7.4 %
6D Rastrigin	91.8	108	15 %
6D Watson	83.4	86.76	3.7 %
9D Hyper-Ellipsoid	73.08	77.48	5.7 %

Table 4. The mean number of PSO iterations for 100 runs for each test function and the corresponding improvement percentage. The proposed algorithm is denoted as NS-PSO.

In Table 3, the mean function evaluations for each test function, as well as the corresponding improvement percentage are given, while in Table 4 the mean number of PSO iterations and the corresponding percentage are given. In both Tables 3 and 4, the proposed algorithm is denoted as NS-PSO.

We shall note here that in the case of the 6-dimensional Rastrigin function, the plain PSO has failed to find the global minimizer in 68% of the cases, while the proposed algorithm has failed only in 16% of the runs. In the Levy No. 3 case the success rate of plain PSO is 75%, while for the proposed algorithm it is 98%. Furthermore, the NSM initialization of the swarm seems to help PSO more in the cases where the bounds of the initial interval are away from the global minimizer. These cases are usually difficult for the PSO technique and result in bad convergence rates and success rates.

5 Conclusions and Further Work

A new idea for initializing the swarm in the PSO technique is presented and experimental results in several test functions are given.

Preliminary results imply that the proposed algorithm can help the PSO to detect faster the promising regions of the search space. Each particle of the initial swarm (except the three first which are randomly selected) is generated performing NSM steps as a vertex of the simplex used. Thus, it possesses an inherent information that guides the swarm in the rest of the optimization faster to the most promising regions. Sometimes it helps even in detecting the global minimizer in cases where the PSO would otherwise fail. The algorithm is not computationally expensive and both NSM and PSO are easily implemented.

Further work includes more sophisticated ways of initialization using the NSM and the development of hybrid versions of PSO, where the main ideas of NSM are applied as operators to the swarm during the optimization.

References:

[1] A.S. Fraser, Simulation of genetic systems by automatic digital computers, *Australian Journal of Biological Science*, Vol.10, 1957, pp. 484-499.

[2] J.H. Holland, *Adaptation in Natural and Artificial Systems*, The MIT Press, 1975.

[3] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1999.

[4] L.J. Fogel, Evolutionary programming in perspective: the topdown view, in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life*, 1994, pp. 135-146.

[5] I. Rechenberg, Evolution Strategy, in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life*, 1994, pp. 147-159.

[6] H.-P. Schwefel, On the evolution of evolutionary computation, in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life*, 1994.

[7] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.

[8] J. Kennedy, R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[9] M. Dorigo, G. Di Caro, The Ant Colony Optimization Meta-Heuristic, in D. Corne, M. Dorigo, and F. Glover (Eds.), *New Ideas in Optimization*, 1999.

[10] E. Bonabeau, C. Meyer, Swarm Intelligence: A Whole New Way to Think About Business, *Harvard Business Review*, Vol.79, No.5, 2001, pp. 106-114.

[11] R.C. Eberhart, J. Kennedy, A new optimizer using Particle Swarm theory, *Proc. Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 2001, pp. 39-43.

[12] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Improving the Particle Swarm Optimizer by Function “Stretching”, in N. Hadjisavvas and P.M. Pardalos (Eds.), *Advances in Convex Analysis and Global Optimization*, 2001, pp. 445-457, Kluwer Academic Publishers.

[13] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Stretching Technique for Obtaining Global Minimizers through Particle Swarm Optimization, *Proc. Particle Swarm Optimization Workshop*, Indianapolis (IN), USA, 2001, pp. 22-29.

[14] K.E. Parsopoulos, M.N. Vrahatis, Modification of the Particle Swarm Optimizer for Locating All the Global Minima, in V. Kurkova, N.C. Steele, R. Neruda, and M. Karny (Eds.), *Artificial Neural Nets and Genetic Algorithms*, 2001, pp. 324-327, Springer.

[15] K.E. Parsopoulos, M.N. Vrahatis, Particle Swarm Optimizer in Noisy and Continuously Changing Environments, in M.H. Hamza (Ed.), *Artificial Intelligence and Soft Computing*, 2001, pp. 289-294, IASTED/ACTA Press.

[16] K.E. Parsopoulos, E.C. Laskari, M.N. Vrahatis, Solving l_1 Norm Errors-in-Variables Problems Using Particle Swarm Optimization, in M.H. Hamza (Ed.), *Artificial Intelligence and*

Applications, 2001, pp. 185-190, IASTED/ ACTA Press.

[17] J.A. Nelder, R. Mead, A simplex method for function minimization, *Computer Journal*, Vol.7, 1965, pp. 308-313.

[18] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes*, Cambridge University Press, 1992.

[19] V. Torczon, On the convergence of the multidirectional search algorithm, *SIAM Journal of Optimization*, Vol.1, 1996, pp. 123-145.

[20] R.C. Eberhart, Y. Shi, Evolving artificial neural networks, *Proc. International Conference on Neural Networks and the Brain*, Beijing, China, 1998, pp. 5-13.

[21] J.J. More, B.S. Garbow, K.E. Hillstrom, Testing Unconstrained Optimization Software, *ACM Transactions on Mathematical Software*, Vol.7, 1965, pp. 308-313.

[22] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, *Journal of Global Optimization*, Vol.11, 1997, pp. 341-359.

[23] D.A. Pierre, *Optimization Theory and Applications*, Dover Publications, 1986.

[24] A. Levy, A. Montalvo, S. Gomez, A. Galderon, *Topics in Global Optimization*, (Lecture Notes in Mathematics No. 909), Springer-Verlag, 1981.