PERGAMON

Nonlinear Analysis 47 (2001) 3419–3424

Nonlinear Analysis

www.elsevier.nl/locate/na

Objective function "stretching" to alleviate convergence to local minima

K.E. Parsopoulos [a,c], V.P. Plagianakos [a,c], G.D. Magoulas [b,c], M.N. Vrahatis [a,c]

[a] *Department of Mathematics, University of Patras, GR-26110 Patras, Greece*

[b] *Department of Information Systems and Computing, Brunel University, Uxbridge UB8 3PH, United Kingdom*

[c] *University of Patras Artificial Intelligence Research Center (UPAIRC)*

Abstract

This paper introduces a new technique for the alleviation of local minima in minimization problems. The proposed "stretching" technique transforms the objective function by stretching upwards the neighborhood of a given point, and assists in eliminating local minima, while preserving the global ones. Experiments indicate that when a global search method converges to an undesired local minimum, the use of the new technique provides a way of escape and helps convergence to the global minimum.

1 Introduction

In many practical optimization problems the search is focused on locating the *global minimizer* x^* of a real-valued objective function $f: \mathcal{E} \to \mathbb{R}$, such that

$$f(x^*) \leq f(x), \quad \forall x \in \mathcal{E}, \tag{1}$$

where the compact set $\mathcal{E} \subseteq \mathbb{R}^D$ is a D-dimensional parallelepiped. There are many Global Optimization (GO) methods developed so far to deal with this problem, which can be classified in two main categories: deterministic methods and probabilistic methods [3,4]. GO strategies possess, in general, strong theoretical convergence properties, and, at least in principle, are straightforward to implement

0362-546X/01/$ - see front matter
PII: S0362-546X(01)00457-6

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. Unfortunately,in practical applications, most of the GO strategies can detect just *sub–optimal solutions* of the function f. In many cases these sub–optimal solutions are acceptable but there are applications where the optimal solution is not only desirable but also indispensable. Therefore, the development of robust and efficient GO methods is a subject of considerable ongoing research.

In this paper a new technique, named *Function "Stretching"* is proposed. It can be combined with GO strategies to help them escaping from local minima by applying a two–step transformation to the objective function. In Section 2, the new technique is presented. In Section 3, the *Particle Swarm Optimizer* (PSO), a recently proposed GO method [2,5], is equipped with the Function "Stretching" technique, and we show through simulation experiments that "Stretching" provides a way of escape from the local minima when PSO's convergence stalls. Finally, concluding remarks are presented in Section 4.

2 "Stretching" the objective function

Let a point $\bar{x}$ be a local minimizer of the objective function f, i.e. there exists a neighborhood B of $\bar{x}$ with $f(\bar{x}) \leq f(x)$, $\forall x \in B$. In many cases, such undesirable minima attract GO methods, resulting premature convergence. To alleviate such problems, soon after a local minimum $\bar{x}$ of the function f has been detected, the following two–step transformation is applied in $f(x)$:

$$\text{Step 1}: \quad G(x) = f(x) + \frac{\gamma_1}{2} \|x - \bar{x}\| \left(\text{sign}(f(x) - f(\bar{x})) + 1\right), \tag{2}$$

$$\text{Step 2}: \quad H(x) = G(x) + \frac{\gamma_2 \left(\text{sign}\left(f(x) - f(\bar{x})\right) + 1\right)}{2 \tanh\left(\mu(G(x) - G(\bar{x}))\right)}, \tag{3}$$

where γ_1, γ_2 and μ are arbitrarily chosen positive constants, and sign$(\cdot)$ defines the well known three–valued sign function.

Step 1 elevates the function $f(x)$ and makes disappear all the local minima which are located above $\bar{x}$. Step 2 helps to stretch the neighborhood of $\bar{x}$ upwards, since it assigns higher function values to those points. Both steps do not alter the local minima located below $\bar{x}$; thus, the location of the global minimum is left unchanged. The effect of this transformation on the form of the objective function is illustrated below for the function *Levy No. 5*:

$$f(x) = \sum_{i=1}^{5} i \cos[(i+1)x_1 + i] \times \sum_{j=1}^{5} j \cos[(j+1)x_2 + j] + \\ +(x_1 + 1.42513)^2 + (x_2 + 0.80032)^2 \tag{4}$$

where $-10 \leq x_i \leq 10, i = 1, 2$. There are about 760 local minima and one global minimum with value $f^* = -176.1375$ at $x^* = (-1.3068, -1.4248)$. The large number of local optimizers makes extremely difficult for any method to locate the global minimizer. In Fig. 1 (left), the original plot of the *Levy No. 5* into the cube $[-2, 2]^2$ is shown. After applying the proposed transformations to the *Levy No. 5* the new form of the function is shown in Fig. 1 (right). The whole neighborhood of the local minimum has been elevated; thus, the former local minimum has now turned to be a local maximum of the function, while lower minima as well as the global one have been left unaffected.

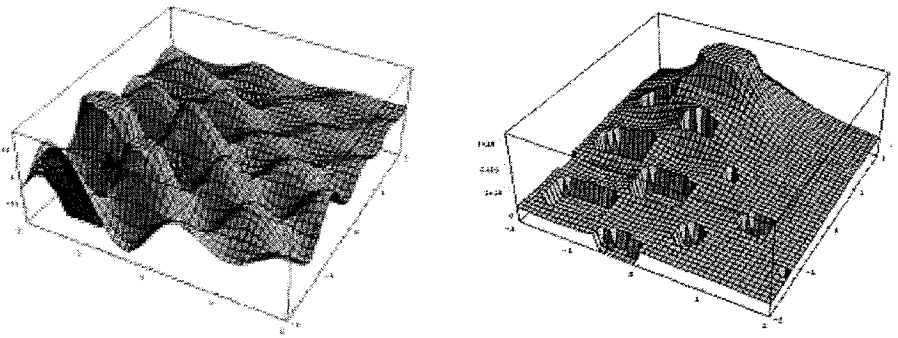

Fig. 1. Plot of the original *Levy No. 5* function (left) and plot of the "stretched" objective function (right).

3 Application example

The proposed technique can be combined with any global search strategy. As an application example, the global search method PSO, which is briefly described below, is equipped with the "Stretching" technique. The modified algorithm is tested in hard-optimization problems, like the minimization of the functions *Levy No. 5* and *Corana*, and in training Artificial Neural Networks (ANNs) for the classification of the classical eXclusive-OR (XOR) patterns.

3.1 Review of the PSO method

In [2,5] Eberhart and Kennedy proposed the PSO algorithm. In PSO the population dynamics do not simulate the human genetic procedures, like other evolution-motivated evolutionary computation techniques, but 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, also called *particles*, during the search for food. Thus, in PSO, each particle (a point in a D–dimensional space) in the population (the set of potential solutions), also called *swarm*, "flies" over the search space looking for promising regions of the landscape. For example, in a minimization problem, promising regions possess lower functional values than previously visited regions. Although, in general, PSO results good solutions, in high–dimensional spaces it stumbles on local minima.

Next, we have used a recently proposed modified PSO [1,7,8] and adopted the following notation: 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 is denoted by the index g. The best previous position (the position giving the best fitness value) of the i-th particle is stored 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 i-th particle's new velocity is calculated using the particle's previous velocity, the distance between the particle's best previous and current position, and the distance between swarm's best experience (the position of the best particle in the swarm) and i-th particle's current position

$$v_{id} = wv_{id} + c_1 r_1(p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id}), \tag{5}$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$ (N is the size of population); c_1 and c_2 are two positive constants; r_1 and r_2 are two random values in the range $[0, 1]$. The *inertia weight* w is employed to control the impact of the previous history of velocities on the current velocity and regulates the trade–off between the global (wide–ranging) and local (nearby) exploration abilities of the swarm. In our case a time decreasing inertia weight has been used: start with a large inertia weight value, to make better global exploration of the search space, and gradually decrease it to get more refined solutions.

In the next step, the i-th particle updates its position by exploiting its own search experience, as well as the experience and the discoveries of other companions according to the relation

$$x_{id} = x_{id} + v_{id}. \tag{6}$$

3.2 Experiments description and results

In all the simulations reported the following parameter values were used: $\gamma_1 = 10000, \gamma_2 = 1$, $\mu = 10^{-10}$, $c_1 = c_2 = 0.5$. Although, our choices seem not to be critical for the success of the method, faster convergence can be obtained by proper fine–tuning. A time decreasing inertia weight, i.e. start from 1 and gradually decrease towards 0.4, worked better than a constant value. In all experiments, when

PSO converges to a local minimum, "Stretching" technique is applied to the original function and PSO is re–initialized.

Table 1
Analysis of the results for the minimization of the Levy No.5 function.

	"Stretching"	No "Stretching"	Overall
Mean Value	3854.2	1049.4	1245.8
Standard Deviation	1630.1	235.1	854.2
Success	7/7	93/93	100/100

The first test refers to the minimization of the *Levy No. 5* function. The results of Table 1 have been obtained after 100 runs with a swarm of size 20, initialized into the cube $[-2, 2]^2$. The average performance is exhibited in terms of the mean value and standard deviation of the number of function evaluations, and successful runs. As it can be seen from the first two columns of the table, in 93 out of 100 cases PSO found the global minimum without any help, while in 7 cases it got stuck in a local minimum. In these cases "Stretching" was applied and the global minimum had finally been detected. Thus the success rate of PSO increased by 7%, reaching an overall success of 100%.

Table 2
Analysis of the results for the minimization of the Corana function.

	"Stretching"	No "Stretching"	Overall
Mean Value	13704.6	2563.2	5460.0
Standard Deviation	7433.5	677.5	6183.8
Success	26/26	74/74	100/100

In the second experiment the *Corana* function is minimized. 100 simulations were run with a swarm of 40 particles, initialized and constrained inside the hypercube $[-1, 1]^4$. Overall success was 100% (see last column of Table 2), but the success of the plain PSO was just 74% (see the No "Stretching" column of Table 2). Thus, the "Stretching" technique increased the success by 26% and significantly improved the performance of the PSO.

Table 3
Analysis of the results for the XOR problem.

	"Stretching"	No "Stretching"	Overall
Mean Value	29328.6	1459.7	7869.6
Standard Deviation	15504.2	1143.1	13905.4
Success	23/23	77/77	100/100

The third experiment concerns training an ANN to classify the XOR patterns. The XOR function maps two binary inputs to a single binary output, and an

ANN with 2 linear input nodes, two hidden nodes with logistic activations and one linear output node was used. From this problem relevant literature, it is known that reaching a global minimizer strongly depends on the initial weights, and that the objective function presents a multitude of local minima. As shown in Table 3, the "Stretching" technique helped to increase significantly the success of the PSO, i.e. from 77% to 100%.

4 Concluding remarks

The main feature of the "Stretching" technique is the use of a two–step transformation of the objective function to eliminate local minima, while preserving the global ones. Experiments indicate that the PSO when equipped with the proposed technique is able to escape from local minima and locate the global one effectively. The "Stretching" technique provides stable convergence, and, thus, a better probability of success for the PSO.

Preliminary experiments with *Evolutionary* and *Genetic* Algorithms [6] show that the "Stretching" technique can assure that these algorithms will not stall to a previously detected minimum. Extensive testing on high–dimensional and more complex real–life optimization tasks is necessary to fully investigate the properties and evaluate the performance of the function "Stretching" technique.

References

[1] R.C. Eberhart and Y.H. Shi, Evolving Artificial Neural Networks, Proc. Int. Conf. on Neural Networks and Brain, Beijing, P.R. China, 1998.

[2] R.C. Eberhart, P.K. Simpson and R.W. Dobbins, Computational Intelligence PC Tools, Academic Press Professional, Boston, MA, 1996.

[3] F. Forgó, Nonconvex Programming, Akadémiai Kiadó, Budapest, 1988.

[4] R. Horst and P.M. Pardalos, Handbook of Global Optimization, Kluwer Academic Publishers, London, 1995.

[5] J. Kennedy and R.C. Eberhart, Particle Swarm Optimization, Proc. IEEE Int. Conf. on Neural Networks, Piscataway, NJ, IV:1942–1948, 1995.

[6] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, New York, 1996.

[7] Y.H. Shi and R.C. Eberhart, Parameter Selection in Particle Swarm Optimization, Proc. Annual Conf. on Evolutionary Programming, San Diego, 1998.

[8] Y.H. Shi and R.C. Eberhart A Modified Particle Swarm Optimizer, Proc. IEEE Int. Conf. on Evolutionary Computation, Anchorage, Alaska, 1998.