Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp117-122)

Particle swarm and simulated annealing for multi-global optimization

A. ISMAEL F. VAZ[1], ANA I.P.N. PEREIRA [2] and EDITE M.G.P. FERNANDES[3]
[1,3]Departamento de Produção e Sistemas, [2]Departamento de Matemática
[1,3]Universidade do Minho, [2]Instituto Politécnico de Bragança
[1,3]Campus de Gualtar, 4710-057 Braga, [2] Campus de Sta Apolónia, 5300 Bragança
PORTUGAL
http://www.norg.uminho.pt

Abstract: - Particle swarm and simulated annealing optimization algorithms proved to be valid in finding a global optimum in the bound constrained optimization context. However, their original versions can only detect one global optimum even if the problem has more than one solution. In this paper we propose modifications to both algorithms. In the particle swarm optimization algorithm we introduce gradient information to enable the computation of all the global and local optima. The simulated annealing algorithm is combined with a stretching technique to be able to compute all global optima. The numerical experiments carried out with a set of well-known test problems illustrate the effectiveness of the proposed algorithms.

Key-Words: - Multi-global optimization, particle swarm optimization, simulated annealing.

1 Introduction

In this paper we address the following optimization problem

$$\min_{x \in X} f(x) \qquad (1)$$

where $f : R^n \to R$ is a given multi-modal objective function and X is a compact set defined by $X = \{x \in R^n : a_j \le x_j \le b_j, j = 1,...,n\}$. A multi-global optimization problem consists of finding all global solutions of problem (1).

So, our purpose is to find all points $x^* \in X$ such that $\forall x \in X, f(x^*) \le f(x)$. Here, we assume that the problem (1) has a finite number of global minimizers and the single function f is continuously differentiable.

This type of problem appears, for example, in chemical engineering, neural networks and in reduction methods for solving semi-infinite programming problems [4], [7], [10], [11]. Due to the existence of multiple local and global optima, these problems cannot be efficiently solved by classical optimization techniques. Recently, Eberhart and Kennedy [3], [9] proposed the particle swarm optimization (PSO) algorithm which is a simple evolutionary algorithm motivated from the simulation of social behavior. Although this is an effective algorithm, when compared with other evolutionary methods, for computing a global solution, some problems can arise when the objective function has more than one global minimum, since the algorithm oscillates between the global minima. To be able to find all global solutions, Parsopoulos and Vrahatis [11] proposed a modification of the PSO algorithm that relies on a function stretching technique, which is used to escape from local minima and to separate the swarm properly whenever a global minimizer is detected.

In this paper, we propose another modification of the PSO algorithm that uses objective gradient information. The ideas behind the stretching technique are also used in the context of a different stochastic method, namely the simulated annealing (SA) method. Although these strategies are quite different, we decided to report on their ability to detect all global solutions of a uni-objective problem.

This paper is organized as follows. We introduce the new PSO and SA algorithms in Sections 2 and 3, respectively. In Section 4 we report some numerical results on a set of test problems to show their efficiency and robustness. Finally, the conclusions make up Section 5.

2 Particle swarm optimization

The particle swarm algorithm mimics a swarm behavior in looking for a certain objective. The PSO algorithm simulates the social behavior concept to compute the global optima of problem (1). In the simplest version, the PSO algorithm should only be applied to problems with at most a global minimum.

To address the problem of computing all the global and local optima we describe in the next subsections the multi-local PSO algorithm which is able to compute all the minima by making use of the objective derivatives.

2.1 Particle swarm optimization algorithm

The PSO algorithm uses a population (swarm) of individuals (particles). To each individual i, at time instant (or iteration) t, is assigned a position $x^i(t) \in X$, and a velocity $v^i(t) \in X$ that provides information to where the individual is traveling to. The velocity at time instant $t+1$ is computed by

$$v_j^i(t+1) = \iota(t)v_j^i(t) + \mu\omega_{1j}(t)(y_j^i(t) - x_j^i(t)) + \nu\omega_{2j}(t)(\hat{y}_j(t) - x_j^i(t)), j = 1,...,n \quad (2)$$

where $\iota(t)$ is the inertial parameter, μ is the cognitive parameter, ν is the social parameter, $\omega_{1j}(t)$ and $\omega_{2j}(t)$ are uniformly random numbers drawn from $(0,1)$, $y^i(t)$ is the best ever position of particle i and $\hat{y}(t)$ is the best ever swarm position. $y_j^i(t) - x_j^i(t)$ is the direction to the particle previous best ever position (cognitive direction) and $\hat{y}_j(t) - x_j^i(t)$ is the direction to the swarm best ever position (social direction). The next velocity is composed by a random combination of the previous velocity with the cognitive and social directions. The new particle position is computed by $x^i(t+1) = x^i(t) + v^i(t+1)$.

2.2 Multi-local particle swarm optimization algorithm

In order to avoid the concentration of all particles around the best swarm position (the global minimum), we propose a multi-local particle swarm optimization (MLPSO) algorithm which differs from the original one, in such a way that the social direction is dropped from equation (2) and the gradient information is used instead. The new equation for the velocity is then

$$v_j^i(t+1) = \iota(t)v_j^i(t) + \mu\omega_{1j}(t)(y_j^i(t) - x_j^i(t)) + \nu\omega_{2j}(t)(-\nabla_j f(y^i(t))). \quad (3)$$

The inclusion of the steepest descent direction in the velocity equation (3) aims to drive each particle to a neighbor local minimum and since we have a population of particles, each one will be driven to a local minimum. Global minima are also detected, since they are local minima as well.

In order for the algorithm to be completely described we need to define the stopping rule. The algorithm terminates when either a specified maximum number of iterations, $N_t^{\max}$, is attained or all the particle have *landed*, i.e., $\max_i \left\| v^i(t+1) \right\|_2 \le \varepsilon_p$.

Including the gradient into the direction can pose some difficulties to the algorithm, since the computed velocity can make particles to get out of the feasible region. To prevent this behavior, the velocity is scaled to fit the maximum velocity allowed and whenever a particle gets out of the feasible region its position is projected onto it.

3 The simulated annealing approach

Usually, the SA method converges to just one global solution in each run. In order to be able to locate all global optima, the SA algorithm is combined with a strategy based on the stretching of the objective function. In the next subsections we briefly describe the main ideas behind a simulated annealing method, the function stretching technique and propose a stretched simulated annealing algorithm.

3.1 Simulated annealing method

The SA method is a stochastic method for global optimization. It can be easily characterized by four main phases: the generation of a new candidate point, the acceptance criterion, the reduction of the control parameters and the stopping rule. The scheme to generate the new candidate point is crucial as it should give a good exploration of the search region and provide a feasible point. Usually, a new point, y, is generated using the current approximation, $x(t)$, and a generating probability density function, $F(x(t))$.

The acceptance criterion allows the SA algorithm to avoid getting stuck in non-global minima, when searching for a global one. Let $A(x(t), c_A(t))$ be the acceptance function which represents the probability of accepting the new point y when $x(t)$ is the current point. The most used criterion in SA algorithms is the Metropolis criterion and is given by

$$x(t+1) = \begin{cases} y & \text{if } \tau \le A(x(t), c_A(t)) \equiv \min\left\{1, e^{-\frac{f(y)-f(x(t))}{c_A(t)}}\right\} \\ x(t) & \text{otherwise,} \end{cases}$$

where τ is a uniformly random number drawn from $(0,1)$. This criterion accepts all new points y such that $f(y) \le f(x(t))$. However, if $f(y) > f(x(t))$, the point y might be accepted with some probability.

The parameter $c_A(t)$ is called the control parameter associated with the acceptance function

and must define a positive decreasing sequence as $t \to \infty$. For a good performance, its initial value must be sufficiently high, so that a good search for promising regions to locate the global minimizer inside the feasible region is accomplished. However, extremely high values yield a slow convergence.

As far as the stopping rule is concerned, the main idea is to stop the process when no further changes occur. Different termination rules are proposed in [1], [2] and [8].

In this work we consider the adaptive simulated annealing (ASA) of Ingber [8]. This variant considers different generating probability density functions for each variable, as in general the objective function behaves differently along different directions. Taking this into consideration, a new feasible candidate point is generated as follows:

$$y_j = x_j(t) + \lambda_j \left(b_j - a_j\right),\ j = 1,...,n,$$

where $\lambda_j \in (-1,1)$ is determined by

$$\lambda_j = sign\left(u - \frac{1}{2}\right)\left(\left(1 + \frac{1}{c_{F_j}(t)}\right)^{|2u-1|} - 1\right) c_{F_j}(t)$$

and u is a uniformly distributed random variable in $(0,1)$. As the process develops, the control parameters $c_{F_j}(t)$ are reduced in order to obtain better approximations to the minimum. Although ASA algorithm claims to converge very quickly, its efficiency depends heavily on a large set of control variables. We refer to [8] for further details on the control parameters $c_{F_j}(t)$ and $c_A(t)$ updating rules.

3.2 The function stretching technique

The function stretching technique is a recently proposed technique [11] that transforms, in a two stage process, the objective function $f(x)$ into a new function $\tilde{f}(x)$. The transformation is carried out along the following lines. After a global minimizer $\bar{x}$ of $f(x)$ has been detected, the first stage of the process elevates the function using

$$\bar{f}(x) = f(x) + \frac{\gamma_1}{2}\|x - \bar{x}\|\left(sign\left(f(x) - f(\bar{x})\right) + 1\right). \quad (4)$$

In the second stage, the neighborhood of $\bar{x}$ is stretched upwards, as follows

$$\tilde{f}(x) = \bar{f}(x) + \gamma_2 \frac{sign\left(f(x) - f(\bar{x})\right) + 1}{2\tanh\left(\xi\left(\bar{f}(x) - \bar{f}(\bar{x})\right)\right)} \quad (5)$$

since higher function values are assigned to its neighbors. In equations (4) and (5), γ_1, γ_2 and ξ are positive constants and $sign(.)$ denotes the well-known sign function.

3.3 Stretched simulated annealing algorithm

We propose a stretched simulated annealing (SSA) approach, which combines the SA algorithm with the function stretching technique previously described. Thus, the SSA algorithm generates a sequence of optimization problems defined as

$$\min_{x \in X} \Phi(x) \equiv \begin{cases} f(x) & if\ t = 1 \\ h(x) & if\ t > 1 \end{cases}$$

where

$$h(x) = \begin{cases} \tilde{f}(x) & if\ x \in V_\varepsilon(\bar{x}) \\ f(x) & otherwise. \end{cases}$$

$\bar{x}$ represents an already detected global minimizer and $V_\varepsilon(\bar{x})$ denotes a neighborhood ε of $\bar{x}$. The sequence of optimization problems are solved by the SA algorithm. As previously shown, $\tilde{f}(x)$ eliminates the already detected global minimum. However, all the minima that are located below and at the same level of $f(\bar{x})$ are not altered, meaning that other global minima will be detected in subsequent iterations. The process stops when either no new global minimum is detected, in a fixed number of successive iterations $N^{\max}$, or a maximum number of function evaluations, $N_{fe}^{\max}$, is reached.

4 Computational experiments

In this section, results from the implementation of the MLPSO and SSA approaches on 32 well-known uni-objective global problems, are reported. The majority of the problems are multi-modal although some difficult uni-modal problems are also included in the test set. In Table 1 we enumerate the test functions and list the number of variables (n), the number of global minimizers (N_{x^*}), and the known global minimum (f^*).

Both algorithms were implemented in the C programming language and connected with AMPL [5] to provide the coded problems. The second column of Table 1 provides the file names of the problems used in the numerical experiments. These problems are well-known in the literature of multi-local and global optimization, and for the sake of

brevity we do not fully describe the problems. The AMPL models can be requested from the first author.

Table 1: Test functions.

	Test functions	n	N_{x^*}	f^*
1	b2	2	1	0
2	bohachevsky	2	1	0
3	branin	2	3	3.979E-01
4	dejoung	3	1	0
5	easom	2	1	-1
6	f1	30	1	-1.257E+04
7	goldprice	2	1	3
8	griewank	6	1	0
9	hartmann3	3	1	-3.863E+00
10	hartmann6	6	1	-3.322E+00
11	hump	2	2	0
12	hump_camel	2	2	-1.0316285
13	levy3	2	18	-1.765E+02
14	parsopoulos	2	12	0
15	rosenbrock10	10	1	0
16	rosenbrock2	2	1	0
17	rosenbrock5	5	1	0
18	shekel10	4	1	-1.054E+01
19	shekel5	4	1	-1.015E+01
20	shekel7	4	1	-1.040E+01
21	shubert	2	18	-1.867E+02
22	storn1	2	2	-4.075E-01
23	storn2	2	2	-1.806E+01
24	storn3	2	2	-2.278E+02
25	storn4	2	2	-2.429E+03
26	storn5	2	2	-2.478E+04
27	storn6	2	2	-2.493E+05
28	zakharov10	10	1	0
29	zakharov2	2	1	0
30	zakharov20	20	1	0
31	zakharov4	4	1	0
32	zakharov5	5	1	0

Initial positions and velocities for the MLPSO and the initial approximation for the SSA algorithms were randomly generated. For each problem, 5 runs have been performed with each technique.

The next two tables report on averaged numbers of: percentage of frequency of occurrence $(f.o.)$, number of iterations (N_t), number of SA calls (N_{SA}), number of function evaluations (N_{fe}), number of gradient evaluations (N_{ge}), best function value (f_m^*), and the best function value attained in the 5 runs (f^*).

For the SSA we set $N^{\max} = 3$, $N_{fe}^{\max} = 100000$, $\gamma_1 = 100$, $\gamma_2 = 1$, $\xi = 10^{-3}$ and $\varepsilon = 0.25$.

For the MLPSO the number of function evaluations is $N_{fe} = s \times N_t$, where s is the swarm size given by $\min(6^n, 1000)$. In the stopping rule $\varepsilon_p = 0.01$ and $N_t^{\max} = 100000$.

In Table 2, we report the results obtained with the MLPSO algorithm.

Table 2: Numerical results obtained by MLPSO.

	$f.o.$	N_t	N_{ge}	f_m^*	f^*
1	100%	68851	1124	1.839E-10	3.143E-11
2	100%	26811	1546	3.808E-11	1.393E-14
3	100%	16386	2425	3.979E-01	3.979E-01
4	100%	14187	45659	5.668E-14	2.603E-16
5	0%	Flat problem			
6	0%	Non differentiable			
7	0%	100000	56	1.206E+02	2.341E+01
8	67%	29873	1217700	5.008E-03	9.745E-09
9	80%	100000	913	-3.789E+00	-3.846E+00
10	0%	100000	3530	-2.901E+00	-3.086E+00
11	100%	24600	996	4.653E-08	4.65E-08
12	100%	22548	944	-1.032E+00	-1.032E+00
13	1%	100000	565	-1.471E+02	-1.722E+02
14	85%	46086	1520	5.144E-17	9.375E-21
15	0%	100000	2126	1.182E+04	7.74E+03
16	0%	100000	46	1.080E+01	1.585E+00
17	0%	100000	3178	3.033E+02	8.516E+01
18	100%	100000	14977	-8.279E+00	-1.008E+01
19	100%	100000	19100	-7.634E+00	-1.001E+01
20	100%	100000	16596	-8.420E+00	-1.003E+01
21	7%	100000	253	-1.422E+02	-1.801E+02
22	100%	24297	3148	-4.075E-01	-4.075E-01
23	90%	68360	451	-1.806E+01	-1.806E+01
24	60%	84587	218	-1.999E+02	-2.278E+02
25	60%	100000	161	-2.277E+03	-2.429E+03
26	40%	100000	4222	-2.388E+04	-2.476E+04
27	10%	100000	46	-1.175E+05	-2.361E+05
28	0%	100000	1829	6.249E+01	4.626E+01
29	100%	21401	3820	1.391E-11	2.948E-14
30	0%	100000	1901	2.016E+02	1.330E+02
31	0%	100000	2362	4.698E+00	2.397E+00
32	0%	100000	1454	9.394E+00	3.132E+00

In Table 3, we report the obtained numerical results with the SSA algorithm.

Both algorithm performances depend on the problem dimension and on the size of the feasible region. In particular, for problems with n greater or equal to 10 both methods failed to detected a global minimum in the specified maximum number of function evaluations, except for problem 28 with the SSA method.

Due to the use of derivative information, the MLPSO algorithm is not able to detect a global minimum in problems with many local minima, as the used swarm size is not large enough.

Although the main goal of this work was to address the multi-global optimization problem, the MLPSO detects global as well as local minima. The SSA algorithm only detects some local minima in particular problems.

5 Conclusions

In this paper, we consider two different stochastic optimization methods for computing all global solutions of a single objective function problem (1). The experiments carried out on a set of test problems show that the simulated annealing algorithm when equipped with the function stretching technique is capable of avoiding local minima and locate the global minimizers with light computational costs and acceptable success rates.

Table 3: Numerical results obtained by SSA.

	$f.o.$	N_{SA}	N_{fe}	f_m^*	f^*
1	100%	5	24066	2.045E-06	3.86E-11
2	100%	6	34411	5.461E-08	1.817E-09
3	100%	6	10529	3.979E-01	3.979E-01
4	100%	4	10606	9.593E-07	9.373E-08
5	100%	4	17422	-1.000E+00	-1.000E+00
6	0%	4	100000	-1.312E+04	-1.336E+04
7	100%	4	26197	3.000E+00	3.000E+00
8	0%	16	100000	1.183E-02	9.858E-03
9	100%	4	13379	-3.863E+00	-3.863E+00
10	100%	5	78301	-3.322E+00	-3.322E+00
11	100%	5	20200	1.349E-07	4.657E-08
12	100%	5	17531	-1.032E+00	-1.032E+00
13	37%	11	18217	-1.765E+02	-1.765E+02
14	100%	15	16542	3.210E-09	2.683E-10
15	0%	4	100000	6.975E-01	7.230E-02
16	80%	10	66902	1.898E-02	1.169E-03
17	60%	6	111073	1.023E-02	6.433E-03
18	80%	7	32961	-1.054E+01	-1.054E+01
19	80%	6	29745	-1.015E+01	-1.015E+01
20	80%	5	22206	-1.040E+01	-1.040E+01
21	99%	32	51684	-1.867E+02	-1.867E+02
22	100%	5	5850	-4.075E-01	-4.075E-01
23	100%	5	39877	-1.806E+01	-1.806E+01
24	100%	5	63510	-2.278E+02	-2.278E+02
25	100%	5	59841	-2.429E+03	-2.429E+03
26	100%	5	101864	-2.478E+04	-2.478E+04
27	100%	5	103191	-2.493E+05	-2.493E+05
28	100%	4	80004	5.765E-03	3.900E-04
29	100%	4	3775	3.368E-07	1.246E-10
30	0%	5	100000	2.567E+00	2.216E+00
31	100%	4	24747	1.807E-06	2.336E-07
32	100%	4	44203	6.72E-06	2.289E-06

While the traditional gradient optimization techniques can be used to compute global and local minima, only one solution can be located in each optimization run. The PSO equipped with the gradient information, as previously shown, is capable of detecting global as well as local minimizers. The heavier computational costs of the MLPSO are balanced by the ability to detect local minimizers.

As future developments we propose to use a derivative-free procedure to generate an approximate descent direction, as proposed in [6], to replace $-\nabla f$ in the velocity equation of the MLPSO algorithm. This procedure will make the algorithm computationally lighter. In order to increase the robustness of the SSA algorithm, in the sense that, some local (non-global) minimizers are also surely detected, we propose to include a strategy that identifies local solutions that satisfy

$$\left| f\left(x^*\right) - f\left(x_i^*\right) \right| < \eta, \quad \text{for } \eta > 0$$

where x_i^* are the desired non-global minimizers and x^* is an already detected global solution.

Acknowledge

Work partially supported by FCT grant POCTI/MAT/58957/2004 and by the Algoritmi research center.

References:

[1] A. Corana, M. Marchesi, C. Martini and S. Ridella, Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm, *ACM Transactions on Mathematical Software*, Vol. 13, No. 3, 1987, pp. 262-280.

[2] A. Dekkers and E. Aarts, Global optimization and simulated annealing, *Mathematical Programming*, Vol. 50, 1991, pp. 367-393.

[3] R. Eberhart and J. Kennedy, New optimizers using particle swarm theory, *Proceedings of the 1995 6th International Sysmposium on Micro Machine and Human Science*, pp. 39-43.

[4] C. Floudas, Recent advances in global optimization for process synthesis, design and control: enclosure all solutions, Computers and Chemical Engineering, 1999, pp. 963-973.

[5] R. Fourer, D.M. Gay and B.W. Kernighan, A modeling language for mathematical programming, *Management Science*, Vol. 36, No. 5, 1990, pp. 519-554. http://www.ampl.com.

[6] A.-R. Hedar and M. Fukushima, Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, *Optimization Methods and Software*, Vol. 19, No. 3-4, 2004, pp. 291-308.

[7] R. Hettich and K.O. Kortanek, Semi-infinite programming: Theory, methods, and applications, *SIAM Review*, Vol. 35, No. 3, 1993, pp. 380-429.

[8] L. Ingber, Adaptive Simulated Annealing (ASA): Lessons Learned, *Control and Cybernetics*, Vol. 25, No. 1, 1996, pp. 33-54.

[9] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp.

1942-1948, Perth, Australia. IEEE Service Center, Piscataway, NJ http://engr.iupui.edu/~shi/ Coference/psopap4.html.

[10] T. León, S. Sanmatías and E. Vercher, A multi-local optimization algorithm, *Top*, Vol. 6, No. 1, 1998, pp. 1-18.

[11] K. Parsopoulos, M. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing*, Vol. 1, 2002, pp. 235-306.

[12] H. Romeijn and R. Smith, Simulated annealing for constrained global optimization, *Journal of Global Optimization*, Vol. 5, 1994, pp. 101-126.