Computers and Mathematics with Applications 57 (2009) 2016–2021

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Learning algorithm for multimodal optimization☆

Xiang Zhao [a,*], Yuan Yao [b], Liping Yan [a]

[a] College of Electronics and Information Engineering, Sichuan University, Chengdu, 610064, China

[b] Department of Electrical and Electronic Engineering, Chengdu University of Information Technology, Chengdu, 610225, China

ARTICLE INFO

Keywords:
Learning algorithm
Multimodal optimization
Evolutionary algorithms

ABSTRACT

We present a new evolutionary algorithm—"learning algorithm" for multimodal optimization. The scheme for reproducing a new generation is very simple. Control parameters, of the length of the list of historical best solutions and the "learning probability" of the current solutions being moved towards the current best solutions and towards the historical ones, are used to assign different search intensities to different parts of the feasible area and to direct the updating of the current solutions. Results of numerical tests on minimization of the 2D Schaffer function, the 2D Shubert function and the 10D Ackley function show that this algorithm is effective and efficient in finding multiple global solutions of multimodal optimization problems.

1. Introduction

For 30 years, optimization procedures have demonstrated outstanding progress. Classic optimization procedures that depend on gradient knowledge or matrix inversion may suffer from numerical instabilities or failure of convergence when the objective function is non-smooth or has discontinuities. In addition they are essentially local optimizations, and consequently not suitable for solving optimization problems with multimodal objective functions. On the other hand, the global optimization techniques such as simulated annealing (SA) [1,2] and genetic algorithms (GA) [3,4] use pseudorandom sampling to search a feasible region, and are able to climb out of the local optima. These global optimization techniques based on random search are also called Monte Carlo techniques. Within a few years of their introduction, they became very popular and were applied in a wide range of areas. Evolutionary algorithms are also notable global optimization techniques [5,6]. They are related to GA but were developed quite independently. In this presentation, we propose a new evolutionary algorithm and call it "learning algorithm". Numerical tests show that this algorithm is effective and efficient for multimodal optimization problems.

2. Learning algorithm

Consider a typical optimization problem (minimization problem) such as

$$\min f(\mathbf{x}), \quad \mathbf{x} \in A \subset \mathbb{R}^n$$

where f is the objective function (or cost function), and $\mathbf{x}$ is a point (i.e. a vector of n dimensions) in its domain of definition (or feasible region) A. Just as with the common evolutionary algorithms, our algorithm searches in A in order to find optimum solutions by reproducing a new generation of solutions at each iterative step. However, our scheme of reproducing the new

☆ The project was supported by the National Natural Science Foundation of China (Grant Nos. 60531010, 60301004).

* Corresponding author.
E-mail addresses: ZhaoXiang59@163.com (X. Zhao), YaoYuan1239@163.com (Y. Yao), Sherry_Yan@163.com (L. Yan).

0898-1221/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2008.10.008

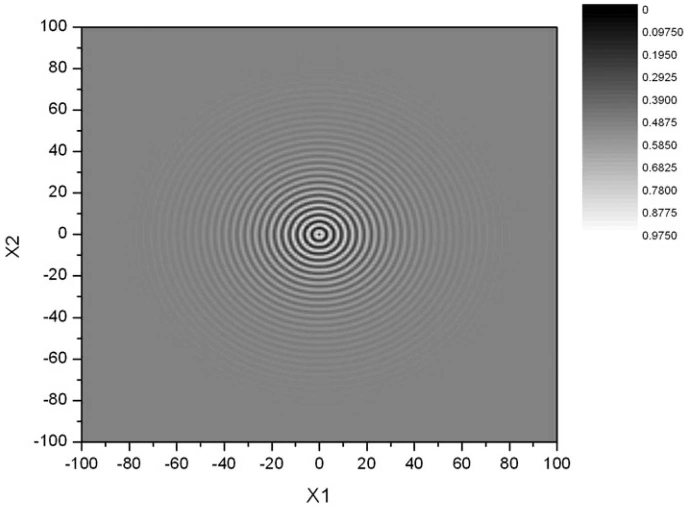

Fig. 1. Schaffer function.

generation is very simple. With this scheme, the most probable areas of the feasible region can be searched more intensively. At each iterative step, two processes, "evaluating cost procedure" and "learning procedure", are performed in turn: (1) costs of all current solutions are evaluated and sorted so that we can choose the top M_{best} solutions and merge them to a list of length M_{best} of historical best solutions; (2) some of the current solutions are perturbed slightly around the historical best solutions, the others may be either moved towards one of the current best solutions or towards one of the historical best solutions or moved randomly in the whole of A. This algorithm can be stated as follows:

- Step 1: Generate $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{M_{\text{ini}}}\} \subset A$ uniformly, evaluate the cost of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{M_{\text{ini}}}$, choose the top M_{best} solutions and insert them into list L,
- Step 2: Generate $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \subset A$ randomly,
- Step 3: Evaluate the cost of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$, choose the top M_{best} solutions $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \ldots, \mathbf{x}_{j_{M_{\text{best}}}}$,
- Step 4: Set $L = L$ merging$\{\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \ldots, \mathbf{x}_{j_{M_{\text{best}}}}\}$ and removing duplicates,
- Step 5: Update $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ according to the following scheme,

    do $i = 1, M$
        if $i \leq M_{\text{best}}$ then
            $\mathbf{x}_i = \mathbf{x}_{j_i} + \mathbf{s}$
        else if $r < P_{\text{learn}}^{(\text{cur})}$ then
            $\mathbf{x}_i = \mathbf{x}_i + r * (\mathbf{x}_{\text{nearest-best}}^{(\text{cur})} - \mathbf{x}_i)$
        else if $r > 1 - P_{\text{learn}}^{(\text{his})}$ then
            $\mathbf{x}_i = \mathbf{x}_i + r * (\mathbf{x}_{\text{nearest-best}}^{(\text{his})} - \mathbf{x}_i)$
        else
            $\mathbf{x}_i = \mathbf{r} \in A$
        end if
    end do

- Step 6: If the termination condition is satisfied, stop; otherwise go to Step 3.

In this algorithm, $\mathbf{s}$ is a random vector having a small enough length, $r \sim U[0, 1]$ a random number, and $\mathbf{r} \sim U(A)$ a random vector. $P_{\text{learn}}^{(\text{cur})}$ and $P_{\text{learn}}^{(\text{his})}$ are called the "learning probability" of the current solution being moved towards the current best solutions and towards the historical ones. Using M_{best}, $P_{\text{learn}}^{(\text{cur})}$ and $P_{\text{learn}}^{(\text{his})}$, learning algorithm can assign different search intensities to different parts of the feasible area and direct the updating of current solutions.

3. Numerical experiments

Here numerical experiments are performed to access the performance of learning algorithm for minimizations of the 2D Schaffer function, the 2D Shubert function and the 10D Ackley function. The three functions are all multimodal functions, and can cause great difficulties for many optimization algorithms. The 2D Schaffer function has a global minimum and high number of local minima around it. In addition, the difference between the values of the local minima and the value of the global minimum is very small (of the order of 10^{-3}). The 2D Shubert function has 760 local minima, 18 of which are global minima. These minima are unevenly spaced. The Ackley function has an exponential term that covers its surface with numerous local minima.