Journal of Computational Science 5 (2014) 224–232

Contents lists available at ScienceDirect

Journal of Computational Science

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

Chaotic bat algorithm

Amir H. Gandomi [a], Xin-She Yang [b,*]

[a] Department of Civil Engineering, The University of Akron, Akron, OH 44325, USA
[b] School of Science and Technology, Middlesex University, Hendon, London NW4 4BT, UK

ARTICLE INFO

Article history:
Received 5 December 2012
Received in revised form 18 July 2013
Accepted 4 October 2013
Available online 14 October 2013

Keywords:
Bat algorithm
Chaos
Metaheuristic
Global optimization

ABSTRACT

Bat algorithm (BA) is a recent metaheuristic optimization algorithm proposed by Yang. In the present study, we have introduced chaos into BA so as to increase its global search mobility for robust global optimization. Detailed studies have been carried out on benchmark problems with different chaotic maps. Here, four different variants of chaotic BA are introduced and thirteen different chaotic maps are utilized for validating each of these four variants. The results show that some variants of chaotic BAs can clearly outperform the standard BA for these benchmarks.

1. Introduction

Many design optimization problems are often highly nonlinear, which can typically have multiple modal optima, and it is thus very challenging to solve such multimodal problems. To cope with this issue, global optimization algorithms are widely attempted, however, traditional algorithms may not produce good results, and latest trends are to use new metaheuristic algorithms [1]. Metaheuristic techniques are well-known global optimization methods that have been successfully applied in many real-world and complex optimization problems [2,3]. These techniques attempt to mimic natural phenomena or social behavior so as to generate better solutions for optimization problem by using iterations and stochasticity [4]. They also try to use both intensification and diversification to achieve better search performance. Intensification typically searches around the current best solutions and selects the best candidate designs, while the diversification process allows the optimizer to explore the search space more efficiently, mostly by randomization [1].

In recent years, several novel metaheuristic algorithms have been proposed for global search. Such algorithms can increase the computational efficiency, solve larger problems, and implement robust optimization codes [5]. For example, Xin-She Yang [6] recently developed a promising metaheuristic algorithm, called bat algorithm (BA). Preliminary studies suggest that the BA can have superior performance over genetic algorithms and particle swarm optimization [6], and it can solve real world and engineering optimization problems [7–10]. On the other hand, recent advances in theories and applications of nonlinear dynamics, especially chaos, have drawn more attention in many fields [10]. One of these fields is the applications of chaos in optimization algorithms to replace certain algorithm-dependent parameters [11].

Previously, chaotic sequences have been used to tune parameters in metaheuristic optimization algorithms such as genetic algorithms [12], particle swarm optimization [13], harmony search [14], ant and bee colony optimization [15,16], imperialist competitive algorithm [17], firefly algorithm [18], and simulated annealing [19]. Such a combination of chaos with metaheuristics has shown some promise once the right set of chaotic maps are used. It is still not clear why the use of chaos in an algorithm to replace certain parameters may change the performance, however, empirical studies indeed indicate that chaos can have high-level of mixing capability, and thus it can be expected that when a fixed parameter is replaced by a chaotic map, the solutions generated may have higher mobility and diversity. For this reason, it may be useful to carry out more studies by introducing chaos to other, especially newer, metaheuristic algorithms.

Therefore, one of the aims of this paper is to introduce chaos into the standard bat algorithm, and as a result, we propose a chaos-based bat algorithm (CBA). As different chaotic maps may lead to different behavior of the algorithm, we then have a set of chaos-based bat algorithms. In these algorithms, we use different chaotic systems to replace the parameters in BA. Thus different methods that use chaotic maps as potentially efficient alternatives to pseudorandom sequences have been proposed. In order to evaluate the

* Corresponding author. Tel.: +44 2084112351.
E-mail addresses: a.h.gandomi@gmail.com, ag72@uakron.edu (A.H. Gandomi), x.yang@mdx.ac.uk (X.-S. Yang).

1877-7503/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2013.10.002

proposed algorithms, a set of unimodal and multimodal mathematical benchmarks are utilized. The simulation results reveal the improvements of the new algorithms, due to the application of deterministic chaotic signals instead of constant values.

The rest of the paper is organized as follows: Section 2 presents the descriptions of the standard BA and proposes four different chaotic BAs. The chaotic maps that generate chaotic sequences in the BA steps are described in Section 3. Section 4 describes how to implement the simulations, while in Section 5, we discuss the tuning of the BA parameters and finding the best chaotic BA. Finally, Section 6 presents the unique features of the chaotic BAs and outlines directions for further research.

2. Bat algorithm

The bat-inspired metaheuristic algorithm, namely the bat algorithm, was recently proposed by Xin-She Yang [6], based on the echolocation of microbats [20]. In the real world, echolocation can have only a few thousandths of a second (up to about 8–10 ms) with a varying frequency in the region of 25–150 kHz, corresponding to the wavelengths of 2–14 mm in the air.

Microbats typically use echolocation for searching for prey. During roaming, microbats emit short pulses; however, when a potential prey is nearby, their pulse emits rates increase and the frequency is tuned up. The increase of the frequency, namely frequency-tuning, together with the speedup of pulse emission will shorten the wavelength of echolocations and thus increase accuracy of the detection. Nature has use frequency-tuning for many years, and in oil industry, they also use frequency tuning to detect different layers of potential oil reserves by increasing the frequency and energy for thinner layers. Bat algorithm was developed to use the key idea of frequency tuning based on the echolocation of microbats. In the standard bat algorithm, the echolocation characteristics of microbats can be idealized as the following three rules:

i. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way;
ii. Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0,1]$, depending on the proximity of their target;
iii. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{\min}$.

The basic steps of BA can be summarized as the pseudo code shown in Fig. 1.

For each bat (say i), we have to define its position $\boldsymbol{x}_i$ and velocity $\boldsymbol{v}_i$ in a d-dimensional search space, and they should be updated subsequently during the iterations. The new solutions $\boldsymbol{x}_i^t$ and velocities $\boldsymbol{v}_i^t$ at time step t can be calculated by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{1}$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{x}_i^{t-1} - \boldsymbol{x}^*)f_i \tag{2}$$

$$\boldsymbol{x}_i^t = \boldsymbol{x}_i^{t-1} + \boldsymbol{v}_i^t \tag{3}$$

where β in the range of [0,1] is a random vector drawn from a uniform distribution. Here $\boldsymbol{x}^*$ is the current global best location (solution) found so far, which is located after comparing all the solutions among all the n bats at the current iteration. As the product $\lambda_i f_i$ is the velocity increment, we can use either f_i (or λ_i) to adjust the velocity change while fixing the other factor λ_i (or f_i),

Bat Algorithm

Objective function $f(\mathrm{x})$, $\mathrm{x} = (x_1, \ldots, x_d)^T$
Initialize the bat population x_i $(i = 1,2, \ldots, n)$ *and* v_i
Define pulse frequency f_i *at* x_i
Initialize pulse rates r *and the loudness* A
while *(t <Max number of iterations)*
  Generate new solutions by adjusting frequency,
  and updating velocities and locations/solutions [equations (2) to (4)]
  if *(rand > r)*
    Select a solution among the best solutions
    Generate a local solution around the selected best solution
  end if
  Generate a new solution by flying randomly
  if *(rand < A & f*$(\mathrm{x}_i) < f(\mathrm{x}_*)$*)*
    Accept the new solutions
  end if
  Rank the bats and find the current best x_*
end while
Postprocess results and visualization

Fig. 1. Pseudo code of the bat algorithm (BA).

depending on the type of the problem of interest. In our implementation, we will use $f_{\min} = 0$ and $f_{\max} = 2$, though the actual range can vary, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency that is drawn uniformly from $[f_{\min}, f_{\max}]$.

By looking at the bat algorithm more closely, we can see that BA can have global and local search abilities, depending on the parameters, and it can also automatically switch from global search to local search by tuning relevant parameters. This switch is controlled by α and γ to be introduce below. The local search is essentially a random walk around the current best solutions, and a new solution for each bat can be generated locally using:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t \tag{4}$$

where the random number ε is drawn from $[-1, 1]$, while $A^t = \langle A_i^t \rangle$ is the average loudness of all the bats at this time step. In fact, this is the main updating equation of simulated annealing. For this reason, simulated annealing can be thought as a very special case of the bat algorithm.

It is worth pointing out that, to a degree, BA can be considered as a balanced combination of local and global moves and this is manifested by controlling the loudness and pulse rate. For simplicity, we can also use $A_0 = 1$ and $A_{\min} = 0$, assuming $A_{\min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound. Now we have

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{5}$$

where α and γ are constants. In fact, α is similar to the cooling factor of a cooling schedule in the simulated annealing. For any $0 < \alpha, \gamma < 1$, we have

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \quad \text{as} \quad t \to \infty \tag{6}$$

In the simplest case, we can use $\alpha = \gamma$, and in the standard BA, we can use $\alpha = \gamma = 0.9$ to 0.975 in most cases, though have used $\alpha = \gamma = 0.9$ in our simulations. However, the main purpose of this paper is to use chaotic maps to tune these 4 parameters so as to see if a chaotic map can improve the efficiency of the bat algorithm. As we can see below, some chaotic maps can indeed enhance the efficient of BA.