Bat Algorithm and Cuckoo Search: A Tutorial

Xin-She Yang

Abstract. Nature-inspired metaheuristic algorithms have attracted much attention in the last decade, and new algorithms have emerged almost every year with a vast, ever-expanding literature. In this chapter, we briefly review two latest metaheuristics: bat algorithm and cuckoo search for global optimization. Bat algorithm was proposed by Xin-She Yang in 2010, inspired by the echolocation of microbats, while cuckoo search was developed by Xin-She Yang and Suash Deb in 2009, inspired by the brood parasitism of some cuckoo species. Both algorithms have shown superiority over many other metaheuristics over a wide range of applications.

1 Bat Algorithm

1.1 Behaviour of Microbats

Bats are fascinating animals. They are the only mammals with wings and they also have advanced capability of echolocation. It is estimated that there are about 1000 different species which account for up to 20% of all mammal species. Their size ranges from tiny bumblebee bats (of about 1.5 to 2 g) to giant bats with a wingspan of about 2 m and weight up to about 1 kg. Microbats typically have a forearm length of about 2.2 to 11 cm [14, 15]. Most bats uses echolocation to a certain degree; among all the species, microbats are a famous example as microbats use echolocation extensively, while megabats do not [1, 5].

Most microbats are insectivores. Microbats use a type of sonar, called echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more

Xin-She Yang
Mathematics & Scientific Computing, National Physical Laboratory,
Teddington TW11 0LW, UK

often use constant-frequency signals for echolocation. The bandwidth of echolocation signals varies with species, and often increases by using more harmonics.

Studies show that microbats use the time delay from the emission and detection of the echo, the time difference between their two ears, and the loudness variations of the echoes to build up three dimensional scenario of the surrounding. They can detect the distance and orientation of the target, the type of prey, and even the moving speed of the prey such as small insects. Indeed, studies suggested that bats seem to be able to discriminate targets by the variations of the Doppler effect induced by the wing-flutter rates of the target insects [1].

1.2 Acoustics of Echolocation

Though each pulse only lasts a few thousandths of a second (up to about 8 to 10 ms), however, it has a constant frequency which is usually in the region of 25 kHz to 150 kHz. The typical range of frequencies for most bat species are in the region between 25 kHz and 100 kHz, though some species can emit higher frequencies up to 150 kHz. Each ultrasonic burst may last typically 5 to 20 ms, and microbats emit about 10 to 20 such sound bursts every second. When hunting for prey, the rate of pulse emission can be sped up to about 200 pulses per second when they fly near their prey. Such short sound bursts imply the fantastic ability of the signal processing power of bats. In fact, studies show the integration time of the bat ear is typically about 300 to 400 μs.

As the speed of sound in air is typically $v = 340$ m/s at room temperature, the wavelength λ of the ultrasonic sound bursts with a constant frequency f is given by

$$\lambda = \frac{v}{f}, \tag{1}$$

which is in the range of 2 mm to 14 mm for the typical frequency range from 25 kHz to 150 kHz. Such wavelengths are in the same order of their prey sizes [1, 14].

Amazingly, the emitted pulse could be as loud as 110 dB, and, fortunately, they are in the ultrasonic region. The loudness also varies from the loudest when searching for prey and to a quieter base when homing towards the prey. The travelling range of such short pulses are typically a few metres, depending on the actual frequencies. Microbats can manage to avoid obstacles as small as thin human hairs.

Obviously, some bats have good eyesight, and most bats also have very sensitive smell sense. In reality, they will use all the senses as a combination to maximize the efficient detection of prey and smooth navigation. However, here we are only interested in the echolocation and the associated behaviour.

Such echolocation behaviour of microbats can be formulated in such a way that it can be associated with the objective function to be optimized, and this makes it possible to formulate new optimization algorithms. We will first outline the basic formulation of the Bat Algorithm (BA) and then discuss its implementation.

1.3 Bat Algorithm

If we idealize some of the echolocation characteristics of microbats, we can develop various bat-inspired algorithms or bat algorithms [18, 20]. For simplicity, we now use the following approximate or idealized rules:

1. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers;
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$ (or wavelength λ), varying wavelength λ (or frequency f) 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;
3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum value $A_{\min}$.

Another obvious simplification is that no ray tracing is used in estimating the time delay and three dimensional topography. Though this might be a good feature for the application in computational geometry; however, we will not use this, as it is more computationally extensive in multidimensional cases.

In addition to these simplified assumptions, we also use the following approximations, for simplicity. In general the frequency f in a range $[f_{\min}, f_{\max}]$ corresponds to a range of wavelengths $[\lambda_{\min}, \lambda_{\max}]$. For example, a frequency range of [20 kHz, 500 kHz] corresponds to a range of wavelengths from 0.7 mm to 17 mm.

For a given problem, we can also use any wavelength for the ease of implementation. In the actual implementation, we can adjust the range by adjusting the frequencies (or wavelengths). The detectable range (or the largest wavelength) should be chosen such that it is comparable to the size of the domain of interest, and then toning down to smaller ranges. Furthermore, we do not necessarily have to use the wavelengths themselves at all, instead, we can also vary the frequency while fixing the wavelength λ. This is because λ and f are related, as λf is constant. We will use this later approach in our implementation.

For simplicity, we can assume $f \in [0, f_{\max}]$. We know that higher frequencies have short wavelengths and travel a shorter distance. For bats, the typical ranges are a few metres. The rate of pulse can simply be in the range of $[0,1]$ where 0 means no pulses at all, and 1 means the maximum rate of pulse emission.

Based on the above approximations and idealization, the basic steps of the Bat Algorithm (BA) can be summarized as the pseudo code shown in Fig. 1.

1.3.1 Movement of Virtual Bats

In the standard bat algorithm [20, 24], we have to use virtual bats. We have to define the rules how their positions x_i and velocities v_i in a d-dimensional search space are updated. The new solutions x_i^t and velocities v_i^t at time step t are given by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{2}$$

```
Bat Algorithm
Initialize a population of n bats $x_i$ $(i = 1,2,...,n)$ and $v_i$
Initialize frequencies $f_i$, pulse rates $r_i$ and the loudness $A_i$
while (t <Max number of iterations)
    Generate new solutions by adjusting frequency,
    and updating velocities and locations/solutions [(2) to (4)]
    if (rand > $r_i$)
        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 < $A_i$ & $f(x_i) < f(x_*)$)
        Accept the new solutions
        Increase $r_i$ and reduce $A_i$
    end if
    Rank the bats and find the current best $x_*$
end while
```

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

$$v_i^{t+1} = v_i^t + (x_i^t - x_*) f_i, \tag{3}$$

$$x_i^{t+1} = x_i^t + v_i^t, \tag{4}$$

where $\beta \in [0,1]$ is a random vector drawn from a uniform distribution. Here x_* is the current global best location (solution) which is located after comparing all the solutions among all the n bats at each iteration t. As the product $\lambda_i f_i$ is the velocity increment, we can use f_i (or λ_i) to adjust the velocity change while fixing the other factor λ_i (or f_i), depending on the type of the problem of interest. In our implementation, we will use $f_{\min} = 0$ and $f_{\max} = O(1)$, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{\min}, f_{\max}]$.

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t, \tag{5}$$

where ε is a random number which can be drawn from a uniform distribution in $[-1,1]$ or a Gaussian distribution, while $A^t = <A_i^t>$ is the average loudness of all the bats at this time step.

The update of the velocities and positions of bats have some similarity to the procedure in the standard particle swarm optimization, as f_i essentially controls the pace and range of the movement of the swarming particles. To a degree, BA can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate.

1.3.2 Loudness and Pulse Emission

Furthermore, the loudness A_i and the rate r_i of pulse emission have to be updated accordingly as the iterations proceed. As the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. For simplicity, we can 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

$$A_i^{t+1} = \alpha A_i^t, \tag{6}$$

and

$$r_i^t = r_i^0 [1 - \exp(-\gamma t)], \tag{7}$$

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

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \ \text{as} \ t \to \infty. \tag{8}$$

In the simplest case, we can use $\alpha = \gamma$, and we have used $\alpha = \gamma = 0.9$ in our simulations.

The choice of parameters requires some experimenting. Initially, each bat should have different values of loudness and pulse emission rate, and this can be achieved by randomization. For example, the initial loudness A_i^0 can typically be around $[1,2]$, while the initial emission rate r_i^0 can be around zero, or any value $r_i^0 \in [0,1]$ if using (7). Their loudness and emission rates will be updated only if the new solutions are improved, which means that these bats are moving towards the optimal solution [18, 20].

1.3.3 Discussions

The bat algorithm is much superior to other algorithms in terms of accuracy and efficiency [18, 25]. If we replace the variations of the frequency f_i by a random parameter and setting $A_i = 0$ and $r_i = 1$, the bat algorithm essentially becomes the standard particle swarm optimization (PSO).

Similarly, if we do not use the velocities, we use fixed loudness and rate: A_i and r_i. For example, $A_i = r_i = 0.7$, this algorithm is virtually reduced to a simple harmony search (HS) [19], as the frequency/wavelength change is essentially the pitch adjustment, while the rate of pulse emission is similar to the harmonic acceptance rate (here with a twist) in the harmony search algorithm. The current studies imply that the proposed new algorithm is potentially more powerful and thus should be investigated further in many applications of engineering and industrial optimization problems.

1.4 Further Topics

Bat algorithms start to attract attention, as many researchers have written to the authors to request a demo code. More applications for both single objective and multiobjective optimization problems have appeared in the literature [25, 24, 16].

From the formulation of the bat algorithm and its implementation and comparison, we can see that it is a very promising algorithm. It is potentially more powerful than particle swarm optimization and genetic algorithms as well as harmony search. The primary reason is that BA uses a good combination of major advantages of these algorithms in some way. Moreover, PSO and harmony search are the special cases of the bat algorithm under appropriate simplifications.

In addition, the fine adjustment of the parameters α and γ can affect the convergence rate of the bat algorithm. In fact, parameter α acts in a similar role as the cooling schedule in the simulated annealing. Though the implementation is slightly more complicated than those for many other metaheuristic algorithms; however, it does show that it utilizes a balanced combination of the advantages of existing successful algorithms with innovative feature based on the echolocation behaviour of bats. New solutions are generated by adjusting frequencies, loudness and pulse emission rates, while the proposed solution is accepted or not, depending on the quality of the solutions controlled or characterized by loudness and pulse rate which are in turn related to the closeness or the fitness of the locations/solution to the global optimal solution.

The exciting results suggest that more studies will be needed to carry out the sensitivity analysis, to analyze the rate of algorithm convergence, and to improve the convergence rate even further. More extensive comparison studies with a more wide range of existing algorithms using much tough test functions in higher dimensions will pose more challenges to all optimization algorithms, and thus such comparisons will potentially reveal the virtues and weakness of all the algorithms of interest.

An interesting extension will be to use different schemes of wavelength or frequency variations instead of the current linear implementation. In addition, the rates of pulse emission and loudness can also be varied in a more sophisticated manner. Another extension for discrete problems is to use the time delay between pulse emission and the echo bounced back. For example, in the travelling salesman problem, the distance between two adjacent nodes/cities can easily be coded as the time delay.

As microbats use time difference between their two ears to obtain three-dimensional information, they can identify the type of prey and the velocity of a flying insect. Therefore, a further natural extension to the current bat algorithm would be to use the directional echolocation and Doppler effect, which may lead to even more interesting variants and new algorithms.

2 Cuckoo Search

Cuckoo search (CS) is one of the latest nature-inspired metaheuristic algorithms, developed in 2009 by Xin-She Yang of Cambridge University and Suash Deb of C. V. Raman College of Engineering. CS was based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights, rather than by simple isotropic random walks. Recent studies showed that CS is potentially far more efficient than PSO and genetic algorithms [21, 22, 3].

2.1 Cuckoo Breeding Behaviour

Cuckoo are fascinating birds, not only because of the beautiful sounds they can make, but also because of their aggressive reproduction strategy. Some species such as the *ani* and *Guira* cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species) [9].

There are three basic types of brood parasitism: intraspecific brood parasitism, cooperative breeding, and nest takeover. Some host birds can engage direct conflict with the intruding cuckoos. If a host bird discovers the eggs are not their owns, they will either get rid of these alien eggs or simply abandon its nest and build a new nest elsewhere. Some cuckoo species such as the New World brood-parasitic *Tapera* have evolved in such a way that female parasitic cuckoos are often very specialized in the mimicry in colour and pattern of the eggs of a few chosen host species. This reduces the probability of their eggs being abandoned and thus increases their reproductivity.

In addition, the timing of egg-laying of some species is also amazing. Parasitic cuckoos often choose a nest where the host bird just laid its own eggs. In general, the cuckoo eggs hatch slightly earlier than their host eggs. Once the first cuckoo chick is hatched, the first instinct action it will take is to evict the host eggs by blindly propelling the eggs out of the nest, which increases the cuckoo chick's share of food provided by its host bird. Studies also show that a cuckoo chick can also mimic the call of host chicks to gain access to more feeding opportunity.

2.2 Lévy Flights

Various studies have shown that the flight behaviour of many animals and insects may pose some typical characteristics of Lévy flights [2, 10]. A recent study showed that fruit flies or *Drosophila melanogaster*, explore their landscape using a series of straight flight paths punctuated by a sudden 90^o turn, leading to a Lévy-flight-style intermittent scale free search pattern [12, 13].

Studies on human behaviour such as the Ju/'hoansi hunter-gatherer foraging patterns also show the typical feature of Lévy flights [4]. Even light can be related to Lévy flights. Subsequently, such behaviour has been applied to optimization and optimal search, and preliminary results show its promising capability [10, 11].

2.3 Cuckoo Search

For simplicity in describing our standard Cuckoo Search developed by Xin-She Yang and Suash Deb [21, 22], we now use the following three idealized rules:

- Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest;
- The best nests with highest quality eggs will be carried over to the next generations;
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0,1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

As a further approximation, this last assumption can be approximated by a fraction p_a of the n host nests are replaced by new nests (with new random solutions). For a maximization problem, the quality or fitness of a solution can simply be proportional to the value of the objective function. Other forms of fitness can be defined in a similar way to the fitness function in genetic algorithms.

For the implementation point of view, we can use the following simple representations that each egg in a nest represents a solution, and each cuckoo can lay only one egg (thus representing one solution), the aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. Obviously, this algorithm can be extended to the more complicated case where each nest has multiple eggs representing a set of solutions, or representing multiobjectives [24].

For this present tutorial, we will use the simplest approach where each nest has only a single egg. In this case, there is no distinction between egg, nest or cuckoo, as each nest corresponds to one egg which also represents one cuckoo.

Based on these three rules, the basic steps of the Cuckoo Search (CS) can be summarized as the pseudo code shown in Fig. 2.

This algorithm uses a balanced combination of a local random walk and the global explorative random walk, controlled by a switching parameter p_a. The local random walk can be written as

$$x_i^{t+1} = x_i^t + s \otimes H(p_a - \varepsilon) \otimes (x_j^t - x_k^t), \tag{9}$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation, $H(u)$ is a Heaviside function, ε is a random number drawn from a uniform distribution, and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$x_i^{t+1} = x_i^t + \alpha L(s,\lambda), \tag{10}$$

where

$$L(s,\lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \tag{11}$$

```
Cuckoo Search via Lévy Flights
Objective function $f(x)$,  $x = (x_1, ..., x_d)^T$
Generate initial population of n host nests $x_i$
while (t <MaxGeneration) or (stop criterion)
   Get a cuckoo randomly/generate a solution by Lévy flights
            and then evaluate its quality/fitness $F_i$
   Choose a nest among n (say, j) randomly
   if ($F_i > F_j$),
      Replace j by the new solution
   end
   Abandon a fraction ($p_a$) of worse nests & generate new solutions
   Keep best solutions (or nests with quality solutions)
   Rank the solutions and find the current best
end while
Postprocess results and visualization
```

Fig. 2 Pseudo code of the Cuckoo Search (CS).

A vectorized implementation can be obtained from this link here[1].

The Lévy flight essentially provides a random walk whose random step length is drawn from a Lévy distribution

$$\text{Lévy} \sim \frac{1}{s^{\lambda+1}}, \quad (0 < \lambda \leq 2), \tag{12}$$

which has an infinite variance with an infinite mean. Here the steps essentially form a random walk process with a power-law step-length distribution with a heavy tail. Some of the new solutions should be generated by Lévy walk around the best solution obtained so far, this will speed up the local search. However, a substantial fraction of the new solutions should be generated by far field randomization and whose locations should be far enough from the current best solution, this will make sure that the system will not be trapped in a local optimum.

The advantages of CS may be related to the characteristics in the algorithm. Firstly, CS is a population-based algorithm, in a way similar to GA and PSO, but it uses some sort of elitism and/or selection similar to that used in genetic algorithms and harmony search. Secondly, the randomization in CS is more efficient, as its step length distribution is heavy-tailed, and any step size (whether large or small) is possible. Thirdly, the number of parameters in CS to be tuned is fewer than GA and PSO, and thus it is potentially more generic to adapt to a wider class of optimization problems. In addition, each nest can have many eggs and thus represent a set of solutions, CS can thus be extended to the type of meta-population algorithms, or even hyper-heuristic algorithms.

[1] http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm

2.4 Choice of Parameters

We have carried out a parametric study by varying the number of host nests (or the population size n), the probability p_a and other parameters. We have used $n = 5, 10, 15, 20, 30, 40, 50, 100, 150, 250, 500$ and $p_a = 0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5$. From our simulations, we found that $n = 15$ to 40, $p_a = 0.25$ to 0.5 and $\lambda = 1$ to 1.5 are sufficient for most optimization problems. In addition, the step size scaling factor α should be linked with the upper limits/bounds U_b and lower bounds L_b in the following empirical way

$$\alpha = 0.01(U_b - L_b), \tag{13}$$

which makes that the steps are not too aggressive (jumping out of the feasible domain), thus ensuring most newly-generated solutions in the right search regions. Here U_b and L_b are d-dimensional vectors with the same dimensions as the solution vector.

Results and analysis also imply that the convergence rate, to some extent, is not sensitive to the parameters used. This means that the fine adjustment is not needed for any given problems.

2.5 How to Do Lévy Flights

Broadly speaking, Lévy flights are a random walk whose step length is drawn from the Lévy distribution, often in terms of a simple power-law formula $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta \leq 2$ is an index. Mathematically speaking, a simple version of Lévy distribution can be defined as

$$L(s, \gamma, \mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp[-\frac{\gamma}{2(s-\mu)}] \frac{1}{(s-\mu)^{3/2}}, & 0 < \mu < s < \infty \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where $\mu > 0$ is a minimum step and γ is a scale parameter. Clearly, as $s \to \infty$, we have

$$L(s, \gamma, \mu) \approx \sqrt{\frac{\gamma}{2\pi}} \frac{1}{s^{3/2}}. \tag{15}$$

This is a special case of the generalized Lévy distribution.

In general, Lévy distribution should be defined in terms of Fourier transform

$$F(k) = \exp[-\alpha |k|^{\beta}], \quad 0 < \beta \leq 2, \tag{16}$$

where α is a scale parameter. The inverse of this integral is not easy, as it does not have analytical form, except for a few special cases.

For the case of $\beta = 2$, we have

$$F(k) = \exp[-\alpha k^2], \tag{17}$$

whose inverse Fourier transform corresponds to a Gaussian distribution. Another special case is $\beta = 1$, and we have

$$F(k) = \exp[-\alpha |k|], \tag{18}$$

which corresponds to a Cauchy distribution

$$p(x, \gamma, \mu) = \frac{1}{\pi} \frac{\gamma}{\gamma^2 + (x-\mu)^2}, \tag{19}$$

where μ is the location parameter, while γ controls the scale of this distribution.

For the general case, the inverse integral

$$L(s) = \frac{1}{\pi} \int_0^{\infty} \cos(ks) \exp[-\alpha |k|^{\beta}] dk, \tag{20}$$

can be estimated only when s is large. We have

$$L(s) \to \frac{\alpha\, \beta\, \Gamma(\beta) \sin(\pi\beta/2)}{\pi |s|^{1+\beta}}, \quad s \to \infty. \tag{21}$$

Here $\Gamma(z)$ is the Gamma function

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt. \tag{22}$$

In the case when $z = n$ is an integer, we have $\Gamma(n) = (n-1)!$.

Lévy flights are more efficient than Brownian random walks in exploring unknown, large-scale search space. There are many reasons to explain this efficiency, and one of them is due to the fact that the variance of Lévy flights

$$\sigma^2(t) \sim t^{3-\beta}, \quad 1 \le \beta \le 2, \tag{23}$$

increases much faster than the linear relationship (i.e., $\sigma^2(t) \sim t$) of Brownian random walks. It is worth pointing out that a power-law distribution is often linked to some scale-free characteristics, and Lévy flights can thus show self-similarity and fractal behavior in the flight patterns. Here β is exactly the parameter λ used earlier.

From the implementation point of view, the generation of random numbers with Lévy flights consists of two steps: the choice of a random direction and the generation of steps which obey the chosen Lévy distribution. The generation of a direction should be drawn from a uniform distribution, while the generation of steps is quite tricky. There are a few ways of achieving this, but one of the most efficient and yet straightforward ways is to use the so-called Mantegna algorithm for a symmetric Lévy stable distribution [8]. Here 'symmetric' means that the steps can be positive and negative.

A random variable U and its probability distribution can be called stable if a linear combination of its two identical copies (or U_1 and U_2) obeys the same distribution. That is, $aU_1 + bU_2$ has the same distribution as $cU + d$ where $a, b > 0$ and

$c, d \in \Re$. If $d = 0$, it is called strictly stable. Gaussian, Cauchy and Lévy distributions are all stable distributions.

In Mantegna's algorithm, the step length s can be calculated by

$$s = \frac{u}{|v|^{1/\beta}}, \tag{24}$$

where u and v are drawn from normal distributions. That is

$$u \sim N(0, \sigma_u^2), \tag{25}$$

and

$$v \sim N(0, \sigma_v^2), \tag{26}$$

where

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\,\beta\,2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1. \tag{27}$$

This distribution (for s) obeys the expected Lévy distribution for $|s| \geq |s_0|$ where s_0 is the smallest step. In principle, $|s_0| \gg 0$, but in reality s_0 can be taken as a sensible value such as $s_0 = 0.1$ to 1.

Studies show that Lévy flights can maximize the efficiency of resource searches in uncertain environments. In fact, Lévy flights have been observed among foraging patterns of albatrosses and fruit flies, and spider monkeys. In addition, Lévy flights have many applications. Many physical phenomena such as the diffusion of fluorescent molecules, cooling behavior and noise could show Lévy-flight characteristics under the right conditions.

The literature on cuckoo search is expanding rapidly. There have been a lot of attention and recent studies using cuckoo search with diverse range of applications [7, 17, 26]. Walton et al. improved the algorithm by formulating a modified cuckoo search algorithm [17], while Yang and Deb extended it to multiobjective optimization problems [26]. Durgun and Yildiz applied it to structural design optimization [6]. Interested readers can refer to more advanced literature [22, 23].

At present, metaheuristic algorithms are inspired by some specific features of the successful biological systems such as social insects and birds. Though they are highly successful, however, these algorithms still have room for improvement. In addition to the above open problems, a truly 'intelligent' algorithm is yet to be developed. By learning more and more from nature and by carrying out ever-increasingly detailed, systematical studies, some truly 'smart' self-evolving algorithms will be developed in the future so that such smart algorithms can automatically fine-tune their behaviour to find the most efficient way of solving complex problems. As an even bolder prediction, maybe, some hyper-level algorithm-constructing metaheuristics can be developed to automatically construct algorithms in an intelligent manner in the not-too-far future.

References

1. Altringham, J.D.: Bats: Biology and Behaviour. Oxford University Press (1996)
2. Barthelemy, P., Bertolotti, J., Wiersma, D.S.: A Lévy flight for light. Nature 453, 495–498 (2008)
3. Bradley, D.: Novel 'cuckoo search algorithm' beats particle swarm optimization in engineering design (news article). In: Science Daily, May 29 (2010); Also in: Scientific Computing (magazine) (June 1, 2010)
4. Brown, C., Liebovitch, L.S., Glendon, R.: Lévy flights in Dobe Ju/'hoansi foraging patterns. Human Ecol. 35, 129–138 (2007)
5. Colin, T.: The Variety of Life. Oxford University Press (2000)
6. Durgun, I., Yildiz, A.R.: Structural design optimization of vehicle components using cuckoo search algorithm. Materials Testing 3, 185–188 (2012)
7. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Cuckoo search algorithm: a meteheuristic approach to solve structural optimization problems. In: Engineering with Computers, July 29 (2011), doi:10.1007/s00366-011-0241-y
8. Mantegna, R.N.: Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Physical Review E 49, 4677–4683 (1994)
9. Payne, R.B., Sorenson, M.D., Klitz, K.: The Cuckoos. Oxford University Press (2005)
10. Pavlyukevich, I.: Lévy flights, non-local search and simulated annealing. J. Computational Physics 226, 1830–1844 (2007)
11. Pavlyukevich, I.: Cooling down Lévy flights. J. Phys. A: Math. Theor. 40, 12299–12313 (2007)
12. Reynolds, A.M., Frye, M.A.: Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search. PLoS One 2, e354 (2007)
13. Reynolds, A.M., Rhodes, C.J.: The Lévy flight paradigm: random search patterns and mechanisms. Ecology 90, 877–887 (2009)
14. Richardson, P.: Bats. Natural History Museum, London (2008)
15. Richardson, P.: The secrete life of bats, `http://www.nhm.ac.uk`
16. Tsai, P.W., Pan, J.S., Liao, B.Y., Tsai, M.J., Istanda, V.: Bat algorithm inspired algorithm for solving numerical optimization problems. Applied Mechanics and Materials 148-149, 34–137 (2012)
17. Walton, S., Hassan, O., Morgan, K., Brown, M.R.: Modified cuckoo search: a new gradient free optimization algorithm. Chaos, Solitons & Fractals 44(9), 710–718 (2011)
18. Yang, X.-S.: A New Metaheuristic Bat-Inspired Algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) NICSO 2010. SCI, vol. 284, pp. 65–74. Springer, Heidelberg (2010)
19. Yang, X.-S.: Harmony Search as a Metaheuristic Algorithm. In: Geem, Z.W. (ed.) Music-Inspired Harmony Search Algorithm. SCI, vol. 191, pp. 1–14. Springer, Heidelberg (2009)
20. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press, UK (2010)
21. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBic 2009), pp. 210–214. IEEE Publications, USA (2009)
22. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Modelling & Numerical Optimisation 1, 330–343 (2010)
23. Yang, X.S.: Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley and Sons, USA (2010)

24. Yang, X.S.: Bat algorithm for multi-objective optimisation. Int. J. Bio-Inspired Computation 3, 267–274 (2011)
25. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Engineering Computations 29(4) (in press, 2012)
26. Yang, X.S., Deb, S.: Multiobjective cuckoo search for design optimization. Computers and Operations Research, October 2011 (2012) (accepted), doi:10.1016/j.cor.2011.09.026