# Cuckoo Search Algorithm: An Introduction

**Presentation** · April 2020



**1 author:**

Xin-She Yang
Middlesex University, UK

# Cuckoo Search: An Introduction

Xin-She Yang

Middlesex University London

For details, please read my book:

**Nature-Inspired Optimization Algorithms**, Elsevier, (2014).

Matlab codes are downloadable from
https://uk.mathworks.com/matlabcentral/profile/authors/3659939-xs-yang

# Almost Everything is Optimization

## Almost everything is optimization ... or needs optimization ...

- Maximize efficiency, accuracy, profit, performance, sustainability, ...
- Minimize costs, wastage, energy consumption, travel distance/time, $CO_2$ emission, impact on environment, ...

## Mathematical Optimization

Objectives: maximize or minimize $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})]$,

$$\boldsymbol{x} = (x_1, x_2, ..., x_D) \in \mathbb{R}^D,$$

subject to multiple equality and/or inequality design constraints:

$$h_i(\boldsymbol{x}) = 0, \quad (i = 1, 2, ..., M),$$

$$g_j(\boldsymbol{x}) \le 0, \quad (j = 1, 2, ..., N).$$

In case of $m = 1$, it becomes a single-objective optimization problem.

Optimization problems can usually be very difficult to solve, especially large-scale, nonlinear, multimodal problems.

In general, we can solve only 3 types of optimization problems:

- Linear programming
- Convex optimization
- Problems that can be converted into the above two

Everything else seems difficult, especially for large-scale problems.
For example, combinatorial problems tend to be really hard – NP-hard!

## Deep Learning

The objective in deep nets may be convex, but the domain is not convex and it's a high-dimensional problem.

$$\text{Minimize} \quad E(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} \Big[ u_i(\boldsymbol{x}_i, \boldsymbol{w}) - \bar{y}_i \Big]^2,$$

subject to various constraints.

# Key Components for Optimization

# Optimization Techniques

There are a wide spectrum of optimization techniques and tools.

## Traditional techniques

- Linear programming (LP) and mixed integer programming.
- Convex optimization and quadratic programming.
- Nonlinear programming: Newton's method, trust-region method, interior point method, ..., barrier Method, ... etc.

But most real-world problems are not linear or convex, thus traditional techniques often struggle to cope, or simply do not work...

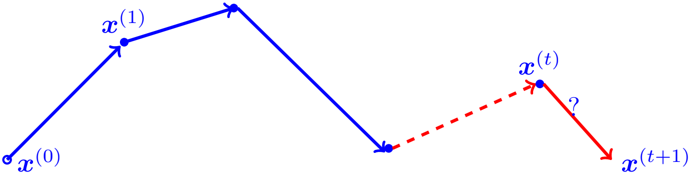
## New Trends – Nature-Inspired Metaheuristic Approaches

- Evolutionary algorithms (evolutionary strategy, genetic algorithms)
- Swarm intelligence (e.g., ant colony optimization, particle swarm optimization, firefly algorithm, cuckoo search, ...)
- Stochastic, population-based, nature-inspired optimization algorithms

# The Essence of an Algorithm

## Essence of an Optimization Algorithm

To generate a better solution point $\boldsymbol{x}^{(t+1)}$ (a solution vector) from an existing solution $\boldsymbol{x}^{(t)}$. That is, $\boldsymbol{x}^{(t+1)} = A(\boldsymbol{x}^{(t)}, \alpha)$ where $\alpha$ is a set of parameters.




Population-based algorithms use multiple, interacting paths.

## Different algorithms

Different ways for generating new solutions!

# Main Problems with Traditional Algorithms

## What's Wrong with Traditional Algorithms?

- Traditional algorithms are mostly local search, thus they cannot guarantee global optimality (except for linear and convex optimization).
- Results often depend on the initial starting points (except linear and convex problems). Methods tend to be problem-specific (e.g., $k$-opt, branch and bound).
- Struggle to cope problems with discontinuity.

## Nature-Inspired Optimization Algorithms

Heuristic or metaheuristic algorithms (e.g., ant colony optimization, particle swarm optimization, firefly algorithm, bat algorithm, cuckoo search, differential evolution, flower pollination algorithm, etc) tend to be a global optimizer so as to

- Increase the probability of finding the global optimality (as a global optimizer)
- Solve a wider class of problems (treating them as a black-box)
- Draw inspiration from nature (e.g., swarm intelligence)

But they can be potentially more computationally expensive.

# Cuckoo Search

Cuckoo search (CS) was developed by Xin-She Yang and Suash Deb in 2009.

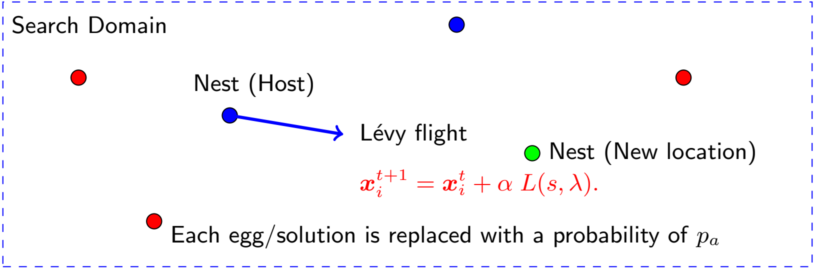
## Cuckoo brood parasitism

- 59 cuckoo species (among 141 cuckoo species) engage the so-called obligate reproduction parasitism strategy.
- Cuckoos lay eggs in the nests of host birds (such as warblers) and let host birds raise their chicks.
- Eggs may be discovered/abandoned with a probability ($p_a \approx 0.25$).
- Co-evolutionary arms race between cuckoo species and host species.

Cuckoo Behaviour (BBC Video)

## Cuckoos' Behaviour and Idealization (Yang and Deb, 2009)

- Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest.
- The best nests with high-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 elsewhere.




Here, $\boldsymbol{x}_i$ is the solution vector (or position of nest $i$) in the search space at iteration $t$, and $\alpha$ is a scaling factor. $L(s, \lambda)$ is the step size to be drawn from the Lévy distribution with an exponent $\lambda$.

# Cuckoo Search (CS) (Yang and Deb, 2009)

Two search mechanisms in CS: local random walks and global Lévy flights.

**Local random walks:**

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + s \otimes H(p_a - \epsilon) \otimes (\boldsymbol{x}_j^t - \boldsymbol{x}_k^t).$$

[$\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k$ are 3 different solutions, $H(u)$ is a Heaviside function, $\epsilon$ is a random number drawn from a uniform distribution, and $s$ is the step size.

**Global random walks via Lévy flights:**

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha L(s, \lambda), \quad L(s, \lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi \lambda / 2)}{\pi} \frac{1}{s^{1+\lambda}}, \ (s \gg s_0).$$

Generation of new moves by Lévy flights, random walks and elitism.

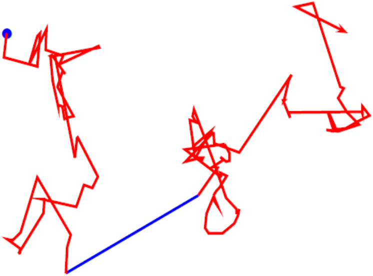
The switch between these two search mechanisms is governed by the discovery probability $p_a = 0.25$.

# Mathematical Foundation for Cuckoo Search

| Isotropic andom walks (diffusion) | Lévy flights (superdiffusion) |
|---|---|
| Gaussian distribution | Lévy distribution |

$$p(s) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(s-\mu)^2}{2\sigma^2}\right], \qquad L(s,\lambda) \sim \frac{1}{\pi}\int_0^\infty \cos(ts) e^{-\alpha t^\lambda} dt.$$

Typical paths of $t = 50$ consecutive steps of random walks



Diffusion distance: $d(t) \sim \sqrt{t}$ $\qquad$ $d(t) \sim t^{(3-\lambda)/2}$ (for $1 \leq \lambda \leq 2$)

## Typical Parameter Values

- Population size: $n = 10$ to $40$ (up to 100 if necessary).
- Lévy exponent: $\lambda = 1.5$.
- $\alpha = O(L/100)$ to $O(L/10)$ where $L$ is the typical scale of the problem. Typically, we can use $\alpha = 0.01$ to $0.1$ for function optimization.
- Number of iterations $t_{\max} = 500$ to $1000$.

## Pseudo-random step size ($s$) for Lévy flights

Quite tricky to generate, though Mantegna's algorithm works well.

$$s = \frac{U}{|V|^{1/\lambda}}, \quad U \sim N(0, \sigma^2), \quad V \sim N(0, 1),$$

where '$\sim$' means 'to draw' random numbers from the probability distribution on the right-hand side. The variance $\sigma^2$ is calculated by

$$\sigma^2 = \Big[ \frac{\Gamma(1+\lambda)}{\Gamma((1+\lambda)/2)} \cdot \frac{\sin(\pi\lambda/2)}{\lambda 2^{(\lambda-1)/2}} \Big]^{1/\lambda},$$

where $\Gamma(\nu)$ is the standard Gamma function. For example, if $\lambda = 1$, we have $\sigma^2 = 1$ since $\Gamma(1+\lambda) = 1$, $\Gamma((1+\lambda)/2) = 1$ and $\sin(\pi/2) = 1$.

# Cuckoo Search Pseudocode

**Algorithm 1:** Cuckoo Search

**Data:** Objective functions $f(\boldsymbol{x})$

**Result:** Best or optimal solution

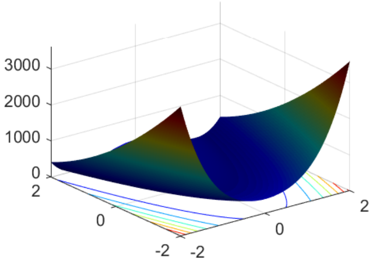
Initialization of parameters ($n$, $p_a$, $\lambda$ and $\alpha$);<br>
Generate initial population of $n$ host nests $\boldsymbol{x}_i$;<br>
**while** *($t$ <MaxGeneration) or (stop criterion)* **do**<br>
    Get a cuckoo randomly;<br>
    Generate a solution by Lévy flights;<br>
    Evaluate its solution quality or objective value $f_i$;<br>
    Choose a nest among $n$ (say, $j$) randomly;<br>
    **if** *($f_i < f_j$)* **then**<br>
        Replace $j$ by the new solution $i$;<br>
    **end**<br>
    A fraction ($p_a$) of worse nests are abandoned;<br>
    New nests/solutions are built/generated;<br>
    Keep best solutions (or nests with quality solutions);<br>
    Rank the solutions and find the current best solution;<br>
    Update $t \leftarrow t + 1$;<br>
**end**

# CS is very efficient

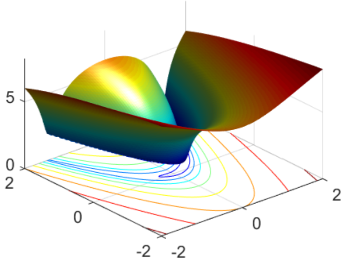
Cuckoo Search Demo: Highly Efficient!

## Rosenbrock (banana) function

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2, \quad (x,y) \in \mathbb{R}^2.$$




$\xrightarrow{\ln[1+f(x,y)]}$




**Cuckoo Search (Demo video at Youtube)** [Please click to start]

# Multi-objective Cuckoo Search (MOCS)

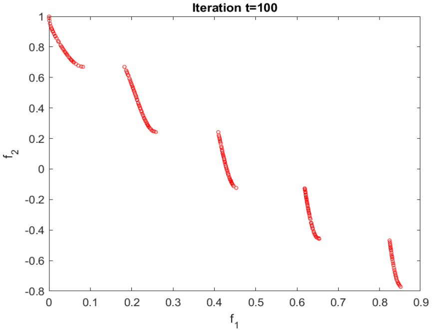
For example, the so-called ZDT function with $D = 30$ dimensions

$$\text{minimize} \quad f_1(\boldsymbol{x}) = x_1, \quad \text{and} \quad f_2(\boldsymbol{x}) = g(\boldsymbol{x})h(\boldsymbol{x}), \quad \boldsymbol{x} \in [0,1]^{30},$$

where

$$g(\boldsymbol{x}) = 1 + \frac{9}{29}\sum_{j=2}^{D=30} x_j, \quad h(\boldsymbol{x}) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1),$$

has a nonconvex Pareto front in the domain $0 \le x_i \le 1$ where $i = 1, 2, ..., 30$.




**Cuckoo Search (Demo video at Youtube)** [Please click to start]

# Cuckoo Search (Demo Codes) and References

## CS Demo Codes

- The standard CS demo in Matlab can be found at the Mathswork File Exchange
  https://uk.mathworks.com/matlabcentral/fileexchange/74767-the-standard-cuckoo-search-cs
- The multi-objective cuckoo search (MOCS) code is also available at
  https://uk.mathworks.com/matlabcentral/fileexchange/74752-multiobjective-cuckoo-search-mocs

## Some References


- Xin-She Yang and Suash Deb, Cuckoo search via Lévy flights, In: *Proceedings of the World Congress on Nature & Biologically Inspired Computing* (NaBIC 2009), IEEE Publications, pp.210-214 (2009).
- Xin-She Yang and Suash Deb, Engineering optimisation by cuckoo search, *Int. J. Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, 330–343 (2010).
- Xin-She Yang and Suash Deb, Multiobjective cuckoo search for design optimization, *Computers & Operations Research*, vol. 40, no. 6, 1616–1624 (2013).
- Xin-She Yang and Suash Deb, Cuckoo search: recent advances and applications, *Neural Computing and Applications*, vol. 24, no. 1, 169–174 (2014).
- Xin-She Yang, Cuckoo Search and Firefly Algorithm: Theory and Applications, Springer, (2013).
- Xin-She Yang, Nature-Inspired Optimization Algorithms, Elsevier Insights, (2014).