Evolutionary Algorithms (EA)

Gergő Bonnyai

July 20, 2022

Contents

List of Figures

Chapter 1

Differential Evolution (DE)

1.1 Story

1.2 Pseudo code

1.3 Flowchart

Chapter 2

Evolutionary Strategy (ES)

2.1 Story

Evolution (or evolutionary) strategy (ES) [1, 2] is a search paradigm inspired by biological evolution. ES implenenting a repeated process of stochastically generating new solution candidates from the actual set of solutions. More precisely in every generation (iteration) the individuals (actual solutions) of the population alone or by forming pairs generate offsprings (new candidates). The fitness of the offsprings are evaluated and the best ones will become the parents for the next generation. There are many variants of evolution strategy. These varians work with different (fitness-based and fitness-independent) mating and recombination strategies. Here I outline a simpler, but most frequently used version where every member of the population on their own generate multiple offsprings. I refer to this process as mutation. At the end of every generation it can be decided to apply elitism where we can keep the best parents in the next generation or not.

Let $X = \{x_1, x_2, \ldots, x_N\}$ population of individuals, where N is the population size and $x_i \in \mathbb{R}^D$. $f : \mathbb{R}^D \to \mathbb{R}^1$ is the fitness function and $fitness_i = f(x_i)$ is the fitness value of x_i. Mutation parameter σ controls the random offspring generation process.
Offsprings are generated using the following formula:

$$z_{ij} = x_i(t) + rand_j * \sigma(t) \tag{2.1}$$

Where $rand_j \in N(0, 1)$, where N stands for normal (Gauss) distribution.
We apply a δ decay parameter on σ to gradually decrease the distance between the parent x_i and the created offspring z_{ij} by generations using the following equation:

$$\sigma(t + 1) = \sigma(t) * \delta \tag{2.2}$$

δ parameter is responsible for the transition between exploration and exploitation phases.

2.2 Pseudo code

Algorithm 1: Evolutionary Strategy

begin
  Set N: population size, T: number of iterations, k: number of offsprings, σ: mutation parameter, δ: decay parameter for σ
  Initialize random population of individuals $X = \{x_1, x_2, \ldots, x_N\}$,
  Calculate fitness values $fitness_i$ for $i \in \{1, 2, ..., N\}$
  while $t \leq T$ *or Stopping criteria not met* **do**
    for $i \leftarrow 1$ *to* N **do**
      for $j \leftarrow 1$ *to* k **do**
        Create offspring z_{ij} by Equation 2.1
        Check search space
        Calculate fitness value $fitness_{ij}$
        if $fitness_{ij} < fitness_{best}$ **then**
          $x_{best} = z_{ij}$
          $fitness_{best} = fitness_{ij}$
        end
      end
    end
    if *apply elitism* **then**
      Create a new population keeping the best N individuals including parents and offsprings based on fitness values.
    end
    else
      Create a new population keeping the best N offsprings based on fitness values.
    end
    Check Stopping Criteria
    Decrease the value of σ by Equation 2.2
    $t = t + 1$
  end
end

2.3 Flowchart

Chapter 3

Cuckoo Search (CS)

3.1 Story

Cuckoo Search (CS) [3] is a metaheuristic algorithm based on the obligate brood parasitic behaviour of some cuckoo species. Cuckoos are birds with an agressive reproduction strategy. Some cockoo species lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Some other cockoo species lay their eggs in the nests of other host birds. Some host birds can engage direct conflict with the intruding cuckoos. Or if a host bird discovers the eggs are not their owns, they will either throw these alien eggs away or simply abandon it's nest and build a new nest elsewhere. In the algorithm in each host nest there is one egg representing a solution. By iterations every cuckoo lays one egg and dumps it's egg in a randomly chosen nest, where a cuckoo egg appears as a new solution. If the new solution is a better one, than it is replaced by the original egg. The best nests will continue in the next iteration. However a given proportion of cuckoo eggs are discovered by the host bird (worst nests) and the host bird throw the egg away/abandon the nest, and build a new nest (new solution) somewhere else. Cuckoo eggs (new solutions) are generated with Lévy flights as researchers found it more efficient in the exploration phase.

Let $X = \{x_1, x_2, \ldots, x_N\}$ population of host nests, where N is the population size and $x_i \in \mathbb{R}^D$. $f : \mathbb{R}^D \to \mathbb{R}^1$ is the fitness function and $fitness_i = f(x_i)$ is the fitness value of x_i.
New cuckoo egg is created by Lévy flight with the following formula:

$$z_i = x_l(t) + rand_i * (x_m - x_n) \tag{3.1}$$

where $rand_i \in$ Lévy(λ), and x_l, x_m, x_n are randomly chosen host nests.

3.2 Pseudo code

Algorithm 2: Cuckoo Search

begin
  Set N: population size, T: number of iterations, k: number of cuckoos, ρ: proportion of abandoned nests
  Initialize random population of host nests $X = \{x_1, x_2, \ldots, x_N\}$,
  Calculate fitness values $fitness_i$ for $i \in \{1, 2, ..., N\}$
  while $t \leq T$ *or Stopping criteria not met* **do**
    for $i \leftarrow 1$ *to* k **do**
      Create cuckoo z_i by Equation 3.1
      Check search space
      Calculate fitness value $fitness_{z_i}$
      Choose a random host nest x_j
      if $fitness_{z_i} < fitness_j$ **then**
        $x_j = z_i$
        $fitness_j = fitness_{z_i}$
      end
    end
    Leave ρ proportion of worst nests and create new random ones.
    Calculate fitness values of newly created host nests.
    Determine x_{best} and $fitness_{best}$.
    Check Stopping Criteria
    $t = t + 1$
  end
end

3.3 Flowchart

Chapter 4

Artificial Bee Colony (ABC)

4.1 Story

Artificial Bee Colony (ABC) [4] algorithm is a swarm based metaheuristic algorithm inspired by the foraging behavior of honey bees. The essential components of the algorithm are the food sources, employed and unemployed foraging bees. The colony of bees is looking for the best food sources. First the bees randomly discover a set of random food sources (population). Then iteratively search for better sources (solutions) applying a special strategy. The colony consists of three type of bees. Employed bees associated with specific food sources, onlooker bees watching the dance of employed bees within the hive to choose a food source, and scout bees searching for food sources randomly. Onlookers and scouts are also called unemployed bees. Initially, all food source positions are discovered by scout bees. Thereafter, the nectar of food sources are exploited by employed bees and onlooker bees, and this continual exploitation will ultimately cause them to become exhausted. Then, the employed bee which was exploiting the exhausted food source becomes a scout bee in search of further food sources once again. In other words, the employed bee whose food source has been exhausted becomes a scout bee. In ABC, the position of a food source represents a possible solution to the problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of employed bees is equal to the number of food sources (solutions) since each employed bee is associated with one and only one food source.

Let $X = \{x_1, x_2, \ldots, x_N\}$ population of food sources, where N is the population size and $x_i \in \mathbb{R}^D$. $f : \mathbb{R}^D \to \mathbb{R}^1$ is the fitness function and $fitness_i = f(x_i)$ is the fitness value of x_i.
Every iteration employed bees search for new food sources using the following

formula:

$$x_{new} = x_i(t) + \phi_i * (x_i(t) - x_{rand}(t)) \tag{4.1}$$

Where $\phi_i \in U(-1, 1)$, where U stands for uniform distribution and $x_{rand}(t)$ is a random food source.
An onlooker bees go to search for new food source after the employed bees shared the food source information with them. If they go in the actual iteration depends on probability. This probability $P_i(t)$ is calculated in every iteration with the following way:

$$P_i(t) = 0.9 * (fitness_i(t)/fitness_{best}(t)) + 0.1 \tag{4.2}$$

The better the $fittness_i(t)$ compared to $fittness_{best}(t)$ the higher the probability of an onlooker bee goes to find a new food source. The onlooker bee uses the same searching strategy as the employed bee. The third type of bees are the scouts who search for a food source randomly. Employed bees whose solutions cannot be improved through a predetermined number of trials called "limit", their solutions are abandoned. Then, the scouts start to search for new solutions, absolutely randomly in the search space.

4.2 Pseudo code

Algorithm 3: Artificial Bee Colony

begin
- Set N: population size, T: number of iterations, k: number of bees, *limit*: number of unsuccessful trials of an employed bee
- Initialize random population of food sources $X = \{x_1, x_2, \ldots, x_N\}$,
- Calculate fitness values $fitness_i$ for $i \in \{1, 2, ..., N\}$
- **while** $t \leq T$ *or Stopping criteria not met* **do**
  - **for** $i \leftarrow 1$ *to* k **do**
    - Employed bee searches for new food source x_{new} by Equation 4.1
    - Check search space
    - Calculate fitness value $fitness_{new}$
    - **if** $fitness_{new} < fitness_i$ **then**
      - $x_i = x_{new}$
      - $fitness_i = fitness_{new}$
    - **end**
    - **else**
      - $bad\_trial_i = bad\_trial_i + 1$
    - **end**
  - **end**
  - **for** $i \leftarrow 1$ *to* k **do**
    - **if** $rand_i < P_i(t)$ **then**
      - Onlooker bee searches for new food source x_{new} by Equation 4.1
      - Check search space
      - Calculate fitness value $fitness_{new}$
      - **if** $fitness_{new} < fitness_i$ **then**
        - $x_i = x_{new}$
        - $fitness_i = fitness_{new}$
      - **end**
    - **end**
    - **else**
      - $bad\_trial_i = bad\_trial_i + 1$
    - **end**
  - **end**
  - **for** $i \leftarrow 1$ *to* k **do**
    - **if** $bad\_trial_i > limit$ **then**
      - Scout bee searches for new random food source x_{new}
      - Calculate fitness value $fitness_{new}$
      - **if** $fitness_{new} < fitness_i$ **then**
        - $x_i = x_{new}$
        - $fitness_i = fitness_{new}$
        - $bad\_trial_i = 0$
      - **end**
    - **end**
  - **end**
  - Check Stopping Criteria
  - $t = t + 1$
- **end**

end

4.3 Flowchart

Chapter 5

Particle Swarm Optimization (PSO)

5.1 Story

5.2 Pseudo code

5.3 Flowchart

Chapter 6

Grey Wolf Optimization (GWO)

6.1 Story

GWO [6] meta-heuristic approach was designed based on the group hierarchy and the hunting strategy of grey wolfs in nature. Grey wolfs normally live in a pack of 5-12 members with strong social hierarchy. The most dominant one, the leader is the alpha (in nature a pair of male and female). The alfa's responsibility to make decisions about hunting, sleeping, etc. The second most dominant is beta, who can be considered as an experienced, skilled member of the pack helping the alpha making decisions. The beta is subordinate to the alfa, but plays a discipliner role for the rest of the wolfes. The omega is the lowest ranked wolf, he has to submit to anyone in the pack. Wolfs who are not alfa, beta or omega are just called subordinates. They play the role of scouts, hunters, sentinels, caretakers. The social behaviour appears in hunting as well in a characteristic fashion. The GWO algorithm mimics the three stage hunting mechanism of grey wolfs: searching for prey, encircling prey and attacking. Four types of pack members can be found in the model. There are three dominant wolfes, alpha, beta and delta. They have the best fitness value. The rest of the wolfs are subordinates or omega who are guided by the three dominant wolfs.

Let $X = \{x_1, x_2, \ldots, x_N\}$ population of wolfs, where N is the population size and $x_i \in \mathbb{R}^D$. $f : \mathbb{R}^D \to \mathbb{R}^1$ is the fitness function and $fitness_i = f(x_i)$ is the fitness value of x_i.
Three parameters are needed to be updated: a, A and C.
a is decreasing from 2 to 0 by iteration linearly:

$$a_{t+1} = 2 * (1 - \frac{t}{T}) \tag{6.1}$$

$$A = 2 * a * rand_1 - a \tag{6.2}$$

So A is a random value in the interval $[-2a, 2a]$.

$$C = 2 * rand_2 \tag{6.3}$$

Hence C is a random value in the interval $[0, 2]$.
Where $rand_i \in U(0, 1)$, where U stands for uniform distribution.

Movement of wolfs determined by the leading wolfs and through coefficient vectors in the following way: $D_\alpha = |C_1 * x_\alpha(t) - x_i(t)|$, $D_\beta = |C_2 * x_\beta(t) - x_i(t)|$, $D_\delta = |C_3 * x_\delta(t) - x_i(t)|$
$X_1 = x_\alpha(t) - A_1 * D_\alpha$, $X_2 = x_\beta(t) - A_2 * D_\beta$, $X_3 = x_\delta(t) - A_3 * D_\delta$

$$x_i(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{6.4}$$

where $x_i(t)$ is the location of the ith wolf at iteration t, and x_α is the location of alpha. $x_\alpha : fitness_\alpha = \min_{i=1,\dots,N} f(x_i)$ (min because of minimization problem). x_β has the second best fitness value, x_δ has the third one.

Searching for prey (exploration) as other phases of hunting guided by the 3 dominant wolf. $|A| > 1$ cases oblige the agent to diverge from the prey and search for better prey (solution). Encircling the prey is also controlled by coefficient vectors A and C and the location of alpha, beta, delta. Attacking of the prey (exploitation) phase is active when $|A| < 1$. In this situation the force towards the prey is getting strong.

6.2 Pseudo code

Algorithm 4: Grey Wolf Optimizer

begin
  Set N: population size, T: number of iterations
  Initialize random population of wolfs $X = \{x_1, x_2, \ldots, x_N\}$,
  Calculate fitness values $fitness_i$ for $i \in \{1, 2, ..., N\}$
  while $t \leq T$ *or Stopping criteria not met* **do**
    Decrease the value of a by Equation 6.1
    Determine the three dominant wolfs $x_{alfa}, x_{beta}, x_{delta}$
    for $i \leftarrow 1$ *to* N **do**
      Update A and C parameters by Equation 6.2 and 6.3
      Update location of wolf x_i by Equation 6.4
      Check search space
      Calculate $fitness_i = f(x_i)$
      if $fitness_i < fitness_{best}$ **then**
        $x_{best} = x_i$
        $fitness_{best} = fitness_i$
      end
    end
    Check Stopping Criteria
    $t = t + 1$
  end
end

Chapter 7

Whale Optimization Algorithm (WOA)

7.1 Story

WOA [5] was inspired by the bubble-net attack of humpback whales. Adult humpback whales have almost the size of a school bus and their main target preys are krills and small fish herds. Whales are very intelligent mammals. They can live and hunt alone and in groups as well. Humpback whales' special hunting method is called bubble-net feeding. This can be observed when small fish herds are close to the surface. The whale dive down first around 12 meters under the herd and then starst moving upward in a spiral shape by creating bubbles along the path to herd the krill herd together before the attack. This manouver was modelled as an optimization algorithm. The formalization is somewhat similar to Grey Wolf Optimizer's. But in this case the agents are drived in the exploitation phase by only one whale with the best fitness. And the exploration and exploitation
Let $X = \{x_1, x_2, \ldots, x_N\}$ population of whales, where N is the population size and $x_i \in \mathbb{R}^D$. $f : \mathbb{R}^D \to \mathbb{R}^1$ is the fitness function and $fitness_i = f(x_i)$ is the fitness value of x_i.
b is a constant parameter. Generally $b = 1$. It affects the spiral encircling move.
4 parameters are needed to be updated: a, A, C and l.
a is decreasing from 2 to 0 by iteration linearly:

$$a_{t+1} = 2 * (1 - \frac{t}{T}) \tag{7.1}$$

$$A = 2 * a * rand_1 - a \tag{7.2}$$

So A is a random value in the interval $[-2a, 2a]$.

$$C = 2 * rand_2 \tag{7.3}$$

Hence C is a random value in the interval $[0, 2]$.

$$l = rand_3 \tag{7.4}$$

Where $rand_1 and rand_2 \in U(0, 1)$, $rand_2 \in U(-1, 1)$, and U stands for uniform distribution.

The exploration and exploitation phases are also controlled by a random mechanism. If $rand < p$ or $rand \geq p$ ($rand \in U(0, 1)$) the algorithm switches between strategies. p is a fixed parameter, generally $p = 0.5$.
The movement of whales in the population determined by the following way:
If $rand < p$ and $|A| < 1$:
$D = |C * x_{best}(t) - x_i(t)|$

$$x_i(t + 1) = x_{best}(t) - A * D \tag{7.5}$$

If $rand < p$ and $|A| > 1$:
$D = |C * x_{rand}(t) - x_i(t)|$

$$x_i(t + 1) = x_{rand}(t) - A * D \tag{7.6}$$

Where x_{rand} is a random member of the whale population.
If $rand \geq p$: $D = |x_{best}(t) - x_i(t)|$

$$x_i(t + 1) = D * exp(bl) * cos(2\pi l) + x_{best}(t) \tag{7.7}$$

where $x_i(t)$ is the location of the ith wolf at iteration t, and x_{best} is the location of the whale with best fitness. $x_{best} : fitness_{best} = \min_{i=1,\ldots,N} f(x_i)$ (min because of minimization problem).

Searching for prey (exploration) as other phases of hunting guided by the 3 dominant wolf. $|A| > 1$ cases oblige the agent to diverge from the prey and search for better prey (solution). Encircling the prey is also controlled by coefficient vectors A and C and the location of alpha, beta, delta. Attacking of the prey (exploitation) phase is active when $|A| < 1$. In this situation the force towards the prey is getting strong.

7.2 Pseudo code

Algorithm 5: Whale Optimization Algorithm

begin
  Set N: population size, T: number of iterations
  Set p: strategy switch probability, b: constant of the spiral
  Initialize random population of whales $X = \{x_1, x_2, \ldots, x_N\}$,
  Calculate fitness values $fitness_i$ for $i \in \{1, 2, ..., N\}$
  while $t \leq T$ *or Stopping criteria not met* **do**
    Decrease the value of a by Equation 7.1
    Determine the best whale x_{best}
    for $i \leftarrow 1$ *to* N **do**
      Update A, C and l parameters by Equation 7.2, 7.3 and 7.4
      if $rand < p$ **then**
        if $|A| < 1$ **then**
          Update location of whale x_i by Equation 7.5
        end
        else
          Update location of whale x_i by Equation 7.6
        end
      end
      else
        Update location of whale x_i by Equation 7.7
      end
      if $fitness_i < fitness_{best}$ **then**
        $x_{best} = x_i$
        $fitness_{best} = fitness_i$
      end
    end
    Check Stopping Criteria
    $t = t + 1$
  end
end

Chapter 8

Flower Pollination Algorithm (FPA)

8.1 Story

8.2 Pseudo code

8.3 Flowchart

Chapter 9

Firefly Algorithm (FA)

9.1 Story

9.2 Pseudo code

9.3 Flowchart

Chapter 10

Black Hole Algorithm (BHA)

10.1 Story

BHA [7, 8] heuristic approach was introduced in 2012. The analogy is to create a random population of stars in the search space, the one with the best fitness value is considered as the black hole. The black hole gives a direction for every star's movement in all iterations. The stars are moving towards the black hole in a random way. After movement if the fitness value of a star is better than the fitness value of the black hole, then this star becomes the black hole. Furthermore another mechanism is involved to make a balance between exploration and exploitation, according to that if a star crosses the event horizon (defined distance from the black hole) then the black hole swallows it. Technically the star loose it's actual position and being redistributed randomly in the search space. Hence a new star is born to keep the population constant.

Let $X = \{x_1, x_2, \ldots, x_N\}$ population of stars, where N is the population size and $x_i \in \mathbb{R}^D$. $f : \mathbb{R}^D \to \mathbb{R}^1$ is the fitness function and $fitness_i = f(x_i)$ is the fitness value of x_i.
Movement of stars towards the black hole:

$$x_i(t+1) = x_i(t) + rand * (x_{BH} - x_i(t)) \tag{10.1}$$

where $x_i(t)$ is the location of the ith star at iteration t, and x_{BH} is the black hole. $x_{BH} : fitness_{BH} = \min_{i=1,\ldots,N} f(x_i)$ (min because of minimization problem). $rand \in U(0,1)$, where U stands for uniform distribution.
Radius of the event horizon is calculated as follows:

$$EventHorizon = \frac{fitness_{BH}}{\sum_{i=1}^{N} fitness_i} \tag{10.2}$$

10.2 Pseudo code

Algorithm 6: Black Hole Algorithm

begin
  Set N: population size, T: number of iterations
  Initialize random population of stars $X = \{x_1, x_2, \ldots, x_N\}$,
  Calculate fitness values $fitness_i$ for $i \in \{1, 2, ..., N\}$
  Determine the black hole x_{BH},
  Calculate $EventHorizon$ by Equation 10.2
  while $t \leq T$ *or Stopping criteria not met* **do**
    for $i \leftarrow 1$ *to* N **do**
      Update location of star x_i by Equation 10.1
      Check search space
      Calculate $fitness_i = f(x_i)$
      if $fitness_i < fitness_{BH}$ **then**
        $x_{BH} = x_i$
        $fitness_{BH} = fitness_i$
        Calculate $EventHorizon$ by Equation 10.2
      end
      else
        if $\|x_{BH} - x_i\| < EventHorizon$ **then**
          Reinitialize x_i randomly within the search space
        end
      end
    end
    Check Stopping Criteria
    $t = t + 1$
  end
end

Bibliography

[1] Rechenberg I., 1971. *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis).* Frommann-Holzboog, 1973

[2] Hans-Paul Schwefel, 1974. *Numerische Optimierung von Computer-Modellen (PhD thesis).* Birkhäuser, 1977

[3] Yang X.-S., Deb S., 2009. *Cuckoo search via Lévy flights.* World Congress on Nature & Biologically Inspired Computing, 2009
Link: https://arxiv.org/abs/1003.1594v1

[4] Karaboga D. 2005. *An idea based on honey bee swarm for numerical optimizations.* Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005

[5] Mirjalili S., Lewis A., 2015. *The Whale Optimization Algorithm.* Elsevier, 2016: p. 51-67.

[6] Mirjalili S., Mirjalili S., Lewis A., 2013. *Grey Wolf Optimizer* Elsevier, 2014: p. 46-61.

[7] Hatamlou, A., 2012. *Black hole: A new heuristic optimization approach for data clustering.* Information sciences, 2012: p. 175-184.

[8] M. Farahmandian, A. Hatamlou, 2015. *Solving optimization problems using black hole algorithm* Journal of Advanced Computer Science & Technology, 2015: p. 68-74.
Link: https://www.sciencepubco.com/index.php/JACST/article/view/4094/1621