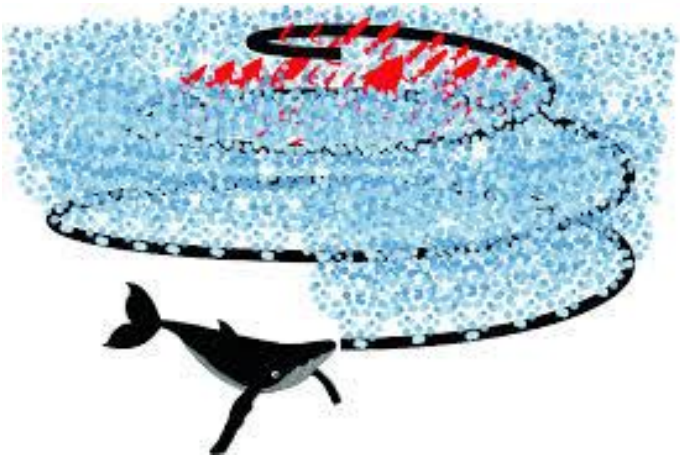
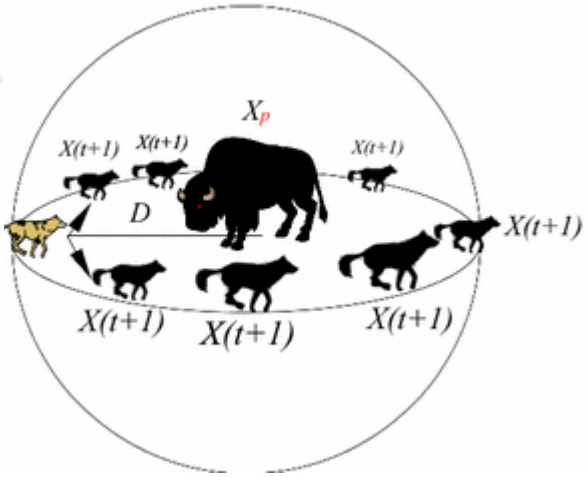
University of Ljubljana, Faculty of Computer and Information Science

Nature inspired metaheuristics





Prof Marko Robnik-Šikonja

Analysis of Algorithms and Heuristic Problem Solving
Edition 2025

Literature

- Fausto, F., Reyna-Orta, A., Cuevas, E., Andrade, Á.G. and Perez-Cisneros, M., 2019. From ants to whales: metaheuristics for all tastes. *Artificial Intelligence Review*, pp.1-58.

Classification of nature inspired methods

- four main categories
  - evolution-based,
  - swarm-based,
  - physics-based,
  - human-based
- **population**-based algorithms, in which a group of randomly generated search agents explore different candidate solutions
- applying a set of rules derived from some **natural phenomenon**
- **interaction** among individuals: a wider knowledge about different solutions,
- **diversity** of the population: efficiently explore the search space, overcome local optima

A general schema of nature inspired methods

- start with a population of solutions x_i, $i \in 1..n$, with dimension d, each assigned a value of objective function $f(x_i)$
- iteratively update current population by modifying some of its properties
- perform selection
- until a stopping criterion is met

Evolution-based methods

- evolution strategies (ES)
- genetic algorithms (GA)
- genetic programming (GP)
- differential evolution (DE)

Swarm-based methods

- simulate the social and collective behavior manifested by groups of animals (such as birds, insects, fishes, and others)
  - particle swarm optimization
  - ant colony optimization
  - artificial bee colony
  - firefly algorithm,
  - social spider optimization
  - …

Physics-based methods

- emulating the laws of physics
  - simulated annealing
  - gravitational search
  - electromagnetism-like mechanism
  - states of matter search

Human-based methods

- draw inspiration from several phenomena commonly associated with humans' behaviors, lifestyle or perception
  - harmony search
  - firework algorithm
  - imperialist competitive algorithm

Evolution-based methods

- evolution strategies (ES)
- genetic algorithms (GA)
- genetic programming (GP)
- differential evolution (DE)

Evolution strategies

- Simple variant
  - $x' = x + N(0, \sigma)$
  - done separately for each dimension
  - selection: select either child or parent, whichever is better
- Several parents:
  - two random parents are chosen
  - each dimension is chosen randomly from one of the parents, the same for σ
  - each dimension is mutated with chosen σ
  - selection

Genetic algorithms

- encoding of genome with a fixed-length representation, e.g., bits
- crossover, mutation, selection

Genetic programming

- variable length encoding
- evolution of programs

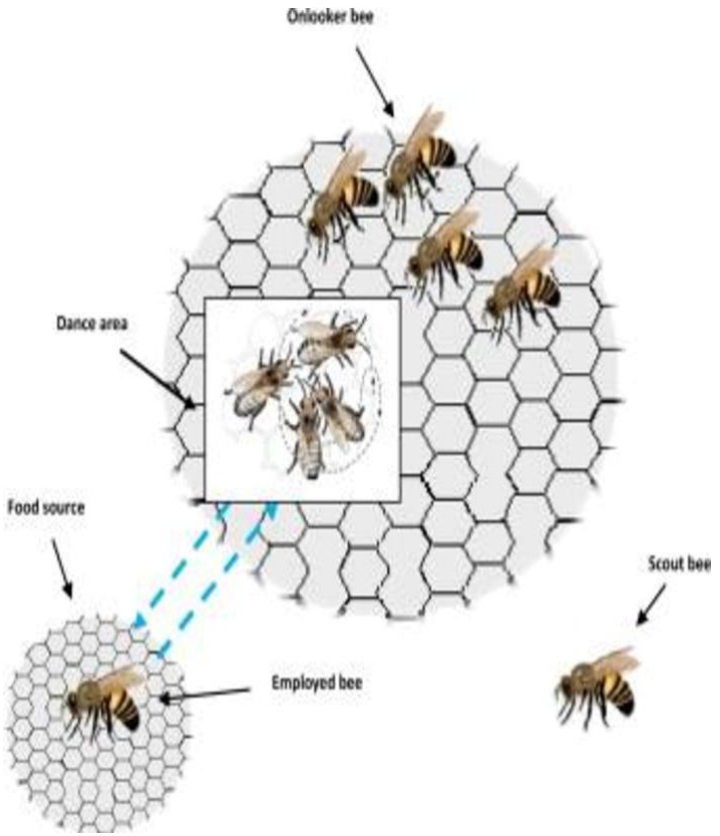
1. Generating an initial population of computer programs, composed by the available functions and terminals (operands).
2. Execute each program in the population and assign it a fitness value according to how well it solves a given problem.
3. Generate a new population of programs by:
   a) Copying the current best computer programs (reproduction).
   b) Creating new offspring programs by randomly changing some parts of a program (mutation).
   c) Creating new offspring programs by recombining parts from two existent programs (crossover).
4. If a specified stop criterion is met, return the single best program in the population as the solution for the pre-specified problem. Otherwise, return to step 2.

Swarm-based methods

- simulate the social and collective behavior manifested by groups of animals (such as birds, insects, fishes, and others)
  - particle swarm optimization
  - ant colony optimization
  - artificial bee colony
  - firefly algorithm,
  - cuckoo search
  - crow search algorithm
  - gray wolf optimizer
  - …

Artificial bee colony (ABC)

- mimicking finding optimal food sources in *d*-dimensional space
- the amount of nectar at each food source is the fitness function
- three types of bees:
  - employed bees,
  - onlooker bees, and
  - scout bees



ABC: Employed bees

- explore the surroundings of individually-known food sources in the hope of finding places with greater amounts of nectar
- share the information of currently known food sources with the rest of the members of the colony
- at each iteration, each employed bee *i* generates a new candidate solution v_i around its currently remembered food source x_i as

$$v_i = x_i + \phi(x_i - x_r)$$

where x_r $(r \neq i)$ is a randomly chosen other food location (sharing the information) and $\phi \in [-1, 1]$ is a randomly selected value

- fitness values of candidate $f(v_i)$ and existing source $f(x_i)$ are compared and the best is kept

- v_i is reminiscent of which other algorithm?

ABC: Onlooker bees

- randomly choose an existing location with probability corresponding to its fitness

$$p_i = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)}$$

- generates a new candidate solution v_i around it, the same as working bee and compare fitness values to select the best

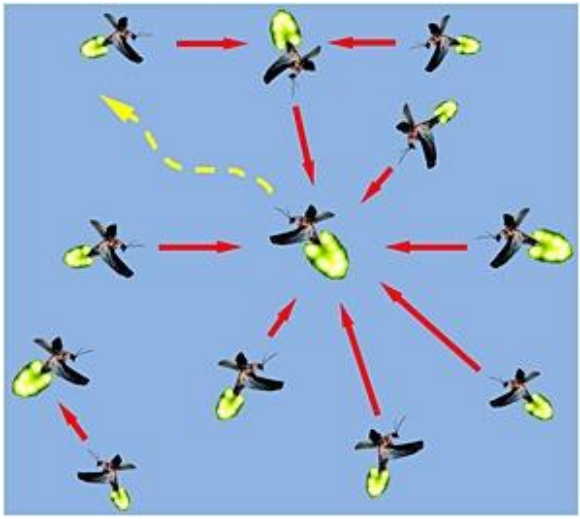
- p_i is reminiscent of which other algorithm?

ABC: Scout bees

- explore the whole terrain for new food sources randomly
- only deployed if a currently known food source is chosen to be abandoned (and thus forgotten by all members of the colony)
- a solution is abandoned if it cannot be improved after a determined number of iterations (parameter *limit)*.
- abandonment assures diversity of solutions during the search process

- ABC:
  - local search: neighborhood exploration and greedy selection of employed and onlooker bees
  - global search: diversification of scout bees

Firefly algorithm



- Xin-She Yang, 2010
- initially, fireflies are positioned randomly
- on each iteration, each firefly jumps toward every other firefly based on how bright it looks
  - the brighter the other firefly appears, the further toward it the selected firefly jumps
  - it only jumps toward fireflies that are brighter than itself
- each firefly shines proportional to its fitness
  - attractiveness falls off with square of the distance
- attractiveness of firefly *i* towards firefly *j* is

$$\beta_{ij} = \begin{Bmatrix} \beta_{0,j} e^{-\gamma r_{i,j}^2} & ; \text{if } f(x_i) > f(x_j) \\ 0 & ; \text{otherwise} \end{Bmatrix},$$

where $\beta_{0,j}$ is attractiveness of firefly *j*, $r_{i,j}$ is theEuclidean distance between fireflies *i* and j, and parameter *γ* is light absorption coefficient, used to vary the overall attractiveness toward the individual j.

- add random jiggling and get the movement

Firefly algorithm - movement

- movement of firefly *i* towards firefly *j* is

$$\Delta x_{ij} = x_i + \beta_{ij}\left(x_j - x_i\right) + \alpha\varepsilon_i$$

ε_i is a *d*-dimensional vector denoting a random movement,

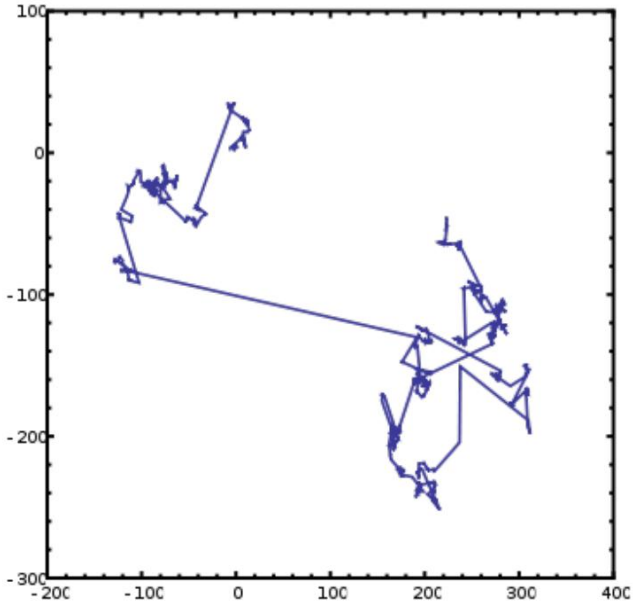
α stands for a randomization parameter with values typically within the interval [0, 1].

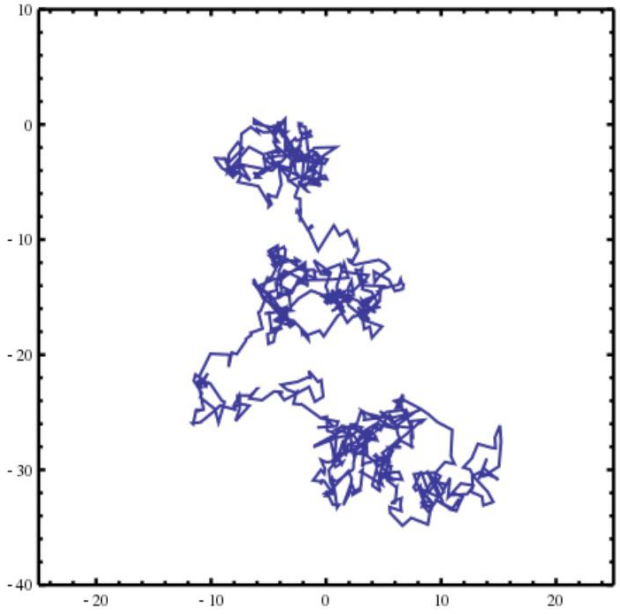
Firefly algorithm – how it works

- The best firefly always stands still, pulling the others toward the best result
- Bad fireflies get pulled in all directions, but good fireflies get pulled much less (this is exploit vs explore)
- Pull diminishes with distance, so the fireflies don’t necessarily all explore the same best position in order to explore more local maxima

Cuckoo search

- (Yang et al, 2010)
- very similar to genetic algorithm
- take candidate, modify it
- if better than existing candidate, replace
- in every generation, discard some proportion of candidates
- fill up with random new ones
- the difference is in how new results are produced
  - using Lévy flights





Cuckoo search

- cuckoos secretly lay their eggs in the nest of other birds in the hope of deceiving the host into thinking that such eggs are their own. Should these alien eggs succeed to be undetected by the host bird, they are almost guaranteed to hatch into new cuckoo chicks; otherwise, the host bird may remove the aliens eggs of their nest or abandon the nest to build a new one somewhere else
- solutions are modeled as a set of *N* host nests
- new candidate solutions are generated around randomly selected host nests *i* by applying a random walk as follows:

$$x_{new} = x_i + \alpha \, \mathrm{Levi}(\lambda)$$

where α is a step factor and λ is a parameter of a random walk

- x_{new} replaces x_i if better
- in each iteration a fraction p_a of worst nest is abandoned and replaced with new randomly generated ones

Cuckoo search

- Balance explore and exploit with Lévy flights
  - usually jump short, sometimes jump long
- Never replace good candidates by bad
  - but always throw away the *n* worst

Crow search algorithm (CSA)

- Askarzadeh, 2016
- inspired by behaviour of flocking crows
- crows hide their excess food at different locations
- crows tend to steal the food from other crows; they follow other crows to their hiding places and steal
- apply different tactics to avoid being stealed from: moving hiding places, trick other crows to follow them to unknown locactions
- a flock of N crows, each with a position x_i in d-dimensional space
- each crow has a memory m_i of its best position so far

Crow search algorithm - movement

- two movement possibilities
  - crow *i* is following a randomly chosen crow *j* to its hiding place
  - crow is tricked into moving into a random position
- At each iteration k, each crow i is moved as follows

$$x_i^{k+1} = \begin{cases} x_i^k + rs_i \cdot fl_i^k \cdot (m_j^k - x_i^k) & \text{if } (\text{rand} \geq \text{AP}_j^k) \\ r_i & \text{if } (\text{rand} < \text{AP}_j^k) \end{cases}$$

crow i not aware of being followed

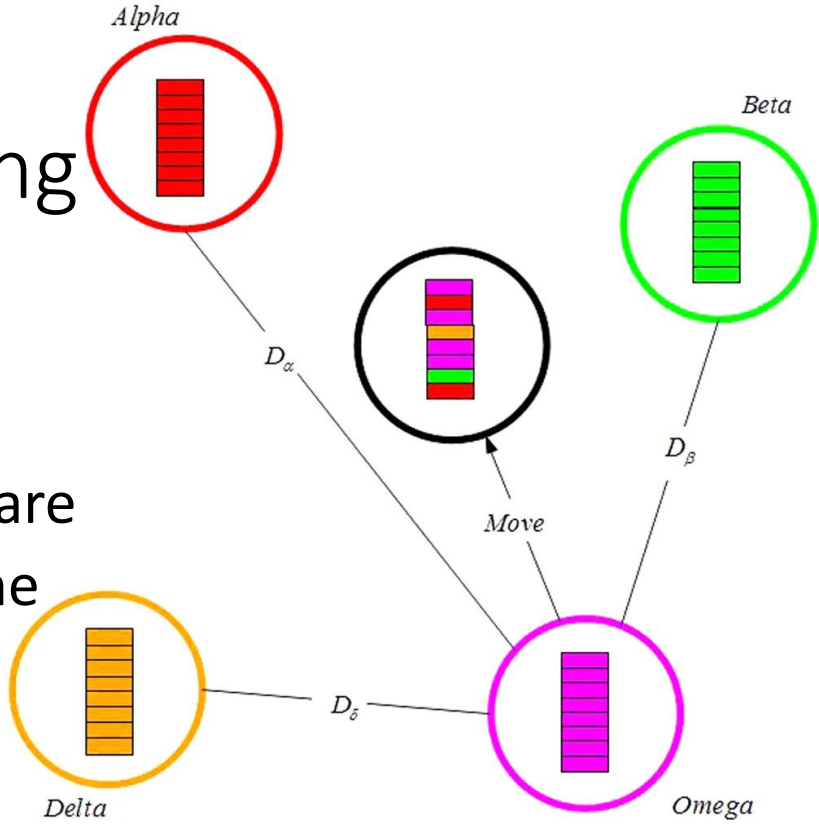
where

- AP_j^k is the awareness probability of crow j at iteration k,
- rand is random number sampled uniformly from [0, 1]
- fl_i^k is the maximum flight length of crow j at iteration k (step size),
- $rs_i \in [0, 1]$ is a random step factor
- r_i is a randomly generated position

Grey wolf optimizer

- (Mirjalili et al., 2014).
- hunting behaviors and social hierarchy observed in packs of grey wolves
- social dominant hierarchy composed of alpha, beta, delta, and omega wolves.
- lower ranked wolves obey higher ranked wolves during hunting
- cooperative hunting tactics in which wolves start to chase an identified prey until it is encircled, and then they proceeded to attack the prey until it is finished
- the fittest solution is alpha wolf (α), the second fittest is beta (β), the third is delta (δ), all the others are omega wolves (ω)
- hierarchy-based search scheme in which lower-ranked wolves move based on information shared by higher-ranked individuals.
- at each iteration, each wolf updates their position as follows

Grey wolf optimizer - moving



- the new position of omega wolf *i*

$x'_i = \frac{\alpha_i + \beta_i + \delta_i}{3}$, where α_i, β_i, and δ_i are position updates for wolf *i* relative to the position of wolfs α, β, and δ (i.e., x_α, x_β, and x_δ):

$$\alpha_i = x_\alpha - A \cdot |Cx_\alpha - x_i|$$
$$\beta_i = x_\beta - A \cdot |Cx_\beta - x_i|$$
$$\delta_i = x_\delta - A \cdot |Cx_\delta - x_i|$$

- here in each iteration we set $A = 2ar_1 - a$ and $C = 2r_2$ where a is a coefficient vector whose values linearly decreases from 2 to 0 over the course of iterations, while r_1 and r_2 are random vector with values uniformly distributed in interval [0, 1].

Physics-based methods

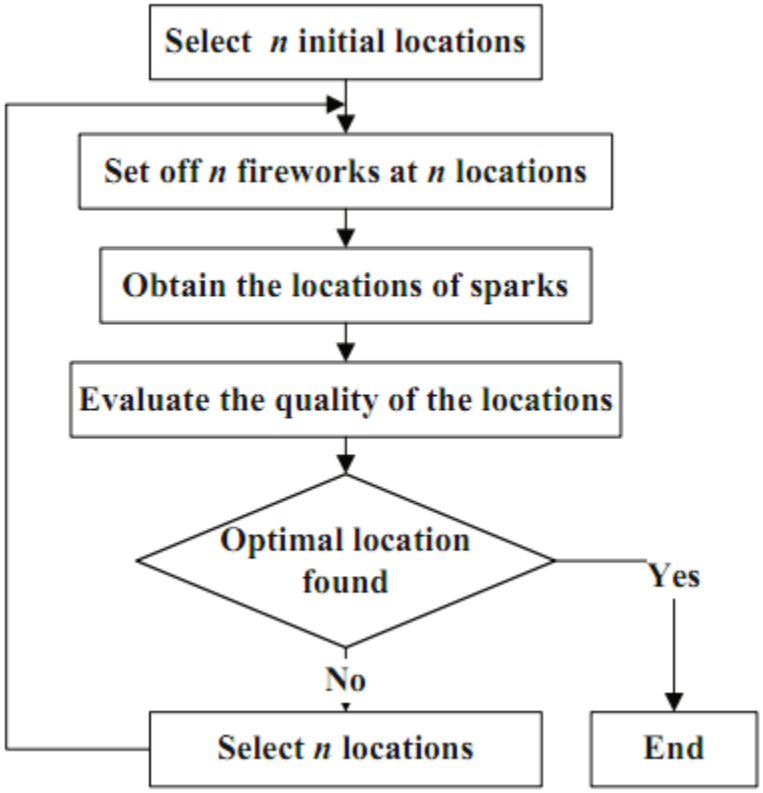
- simulated annealing
- gravitational search
- electromagnetism-like mechanism
- states of matter search

Human-based methods

- harmony search
- firework algorithm
- imperialist competitive algorithm

Fireworks algorithm

- intensification of search in promising directions



Artificial immune systems

- Several algorithms mimicking basic ideas of immune systems
- identification: how to separate self and non-self

Negative selection

- produce a set of self strings, S, that define the normal state of the system.
- the task is to generate a set of detectors, D, that recognise the complement of S (non-self)
- detectors can be applied to new data in order to classify them as being self or non-self
- e.g. detect data manipulation caused by a virus in a computer system

---

```
input   : S_seen = set of seen known self elements
output  : D = set of generated detectors

begin

   repeat
      Randomly generate potential detectors and place them in a set P
      Determine the affinity of each member of P with each member of the self set S_seen
      If at least one element in S recognises a detector in P according to a recognition threshold,
         then the detector is rejected, otherwise it is added to the set of avaliable detectors D
   until  Stopping criteria has been met

end
```

Clonal selection

- computational optimisation and pattern recognition tasks
- e.g., pattern matching and multi-modal function optimization
- a set of patterns, S, to be matched are considered to be antigens; the task of the algorithm CLONALG is to produce a set of memory antibodies, M, that match the members in S.

```
input   : S = set of patterns to be recognised, n the number of worst elements to select for removal
output  : M = set of memory detectors capable of classifying unseen patterns

begin

   Create an initial random set of antibodies, A

   forall  patterns in S   do
      Determine the affinity with each antibody in A
      Generate clones of a subset of the antibodies in A  with the highest affinity.
         The number of clones for an antibody is proportional to its affinity
      Mutate attributes of these clones to the set A , and place a copy of the highest
         affinity antibodies in A  into the memory set, M
      Replace the n lowest affinity antibodies in A with new randomly generated antibodies
   end

end
```

Why so many algorithms?

- Publishing standards for research journals and conferences are probably too lax
- Algorithm descriptions are weak
  - far too little information about tuning parameters
  - often no code available
- Evaluation sections are weak
  - only one evaluation metric
  - no information about how competing baseline methods (e.g., PSO, DE, GA) were tuned
- no information about how a proposed algorithm was tuned
- no cross-comparison with other algorithms