# Artifical Bee Colony Algorithm

## What is ABC?

- **Creator**: Dervis Karaboga, 2005.
- **Inspiration**: The food foraging stratergy of honey bees.
- **Key Idea**: It models how bees find, share information about, and exploit food sources with the highest nectar.
- The "food source" is our solution, and the "nectar" is the fitness or quality of that solution.
- The algorithm divides the population into three types of bees: employed, onlooker, and scout.
- The approach balances exploration, finding new areas, and exploitation, refining good solutions.

## The Bee Colony in Action

- **Employed Bees**: Each employed bee is assigned to a specific food source (a solution). They locally search for a better solution in their neighbourhood.
- **Onlooker Bees**: These bees wait at the hive and decide which food source to exploit based on the waggle dances of the employed bees.
- The more *nectar* an employed bee's source has, the more likey an onlooker will choose it.
- **The Seach Equation**: A new candidate solution, $v_{ij}$ is created for an employed or onlooker bee from its current position $x_{ij}$ and a randomly chosen neighbour $x_kj$ using the formula:

$$v_ij = x_ij + \phi_ij(x_{ij} - x_{kj})$$

- Here $\phi_ij$ is a random number between [-1, 1], controlling the search step size

## What about the Onlooker and Scout Bees?

- **Onlooker bees**: These bees wait at the hive and decide which food source to exploit based on the waggle dances of the employed bees.
- **Selection**: The better a food source's nectar, the higher the probability that an onlooker bee will choose it. The probability is calculated with the fitness of the source:

$$p_i = \frac{fit_i}{\sum_{j=1}^{FN} fit_j}$$

- After choosing a source, the onlooker bee goes to it and performs a local search, similar to an employed bee.
- **Scout Bees**: If an employed bee's food source doesn't improve after a certain number of trials, it's considered abandoned.
- That employed bee then becomes a scout bee, which means it flies off to find a brand new, randomly generated food source.

- This scout phase is critical for preventing the algorithm from getting stuck in a local optimum.
- The overall process is a cycle of employed, onlooker, and scout bee phases, with the best solution found so far always being remembered.

### Flowchart

`Akhil add flowchart`

---

## Spider Monkey Optimisation Algorithm

### Introducing Spider Monkeys and their combined intellect

- **Creator**: J. C. Bansal et al., 2014.
- **Inspiration**: The "fission - fusion" social structure of spider monkey groups as they forage for food.
- **Key Idea**: Monkeys live in a large group, but for foraging, they split into smaller, more manageable subgroups (fission) to reduce competition.
- Later, they come back together (fusion) to share information.
- This is a population-based algorithm that uses this group dynamic to find optimal solutions.

### The Fission - Fusion Process

- The algorithm starts with an initial population of "spider monkeys" (candidate solutions).
- The entire group has a "global leader" who guides the main search.
- The group splits into smaller subgroups, each with a "local leader."
- **Local Leader Phase**: Monkeys in a subgroup update their positions by moving towards their local leader's best position and incorporating information from other random monkeys in the subgroup:

  $$SM_new, ij = SM_ij + rand(0,1)(LL_kj - SM_ij) + rand(-1,1)(SM_rj - SM_ij)$$

- **Global Leader Phase**: The entire swarm is considered one group. All monkeys update their positions based on the best solution found by the entire swarm, the Global Leader:

  $$SM_new, ij = SM_ij + rand(0,1)(GL_j - SM_ij) + rand(-1,1)(SM_rj - SM_ij)$$

### Global Collaboration

- **Global Leader Learning Phase**: The global leader is updated using greedy selectio from the population. If the GL remains the same the GL limit is incremented.

- **Local Leader Learning Phase**: The local leader is updated using greedy selection from the specific groups. If the LL remains the same the LL limit is incremented.
- **Local Leader Decision Phase**: If the LL limit goes beyond a set constant, all monkeys in that group are randomly redistributed with weightage for the GL and LL:

  $SM_new, ij = SM_minj + U(0,1)(SM_maxj - SM_minj) .........ifU(0,1) >= pr$

  $SM_new, ij = SM_ij + U(0,1)(GL_j - SM_ij) + U(0,1)(SM_ij - LL_kj) ............otherwise$
- **Global Leader Decision Phase**: If the GL limit goes about a set a constant, the population is divided.

## Flowchart

`Akhil add flowchart`

---

# Particle Swarm Optimisation

## The Swarm and the Solution

- *Creators*: James Kennedy and Russell C. Eberhart, 1995.
- **Inspiration**: The social behaviour of bird flocks or fish schools.
- **Key Idea**: It models how individuals in a group can find an optimal solution by following the best-performing members of the group.
- Each "particle" in the swarm is a potential solution.
- The search space is multi-dimensional, and each particle "flies" through this space.

## How Particles Fly

- Each particle keeps track of its own best-known position, which we call its personal best, or pbest.
- The entire swarm keeps track of the best-known position found by any particle so far, which is the global best, or gbest.
- The magic is in the velocity update equation. A particle's new velocity is a mix of three things:
  1. Its previous velocity (momentum).
  2. A pull towards its own pbest.
  3. A pull towards the swarm's gbest.
- The equation for velocity update is:

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (pbest_i(t) - x_i(t)) + c_2 r_2 (gbest(t) - x_i(t))$$

### Putting it All Together

- After a new velocity is calculated, the particle updates its position using a simple formula:
$$x_i(t+1) = x_i(t) + v_i(t+1)$$
- This process is repeated over generations.
- The balance between the pbest and gbest terms is crucial. The pbest term gives the particle its individual exploration, while the gbest term provides social cooperation.

$$pbest_i(t+1) = x_i(t+1)............iff(x_i(t)) < f(pbest_i(t))$$

$$pbest_i(t+1) = pbest_i(t)............otherwise$$

$$gbest(t+1) = \arg \min_{pbest_k in all pbests} f(pbest_k)$$

- This simple yet powerful mechanism allows the swarm to collectively converge on an optimal solution.

### Flowchart

`Akhil add flowchart`

---

## Genetic Algorithm

### Evolution in Computation

- **Creator**: John Holland, 1960-1970s.
- **Inspiration**: Charles Darwin's theory of natural evolution and "survival of the fittest."
- **Key Idea**: It applies the principles of natural selection—heredity, mutation, and crossover—to a population of candidate solutions.
- A solution is represented as a "chromosome" (a string of genes, like a binary string).
- The algorithm iteratively improves the population by selecting the "fittest" individuals to produce a new generation.

### The Three Core Operators

- **Selection**: The "fittest" individuals (solutions with high-quality fitness scores) are selected from the current population to act as "parents."
- This is often done using methods like roulette wheel selection, where better solutions have a higher probability of being chosen.
- **Crossover**: Pairs of parents are chosen, and a "crossover" point is selected. Their genetic material (the solution string) is swapped to create new "offspring."

- For example, in a one-point crossover on binary strings, if Parent 1 is 110011 and Parent 2 is 001100, they could produce offspring 111100 and 000011.
- **Mutation**: After crossover, a small, random change is introduced to the offspring's genes. This is vital for maintaining diversity and preventing premature convergence.
- For a binary string, this might mean flipping a 0 to a 1 or vice-versa.

### The Evolutionary Loop

- The process starts with an initial, randomly generated population.
- The population is evaluated for fitness, and the cycle of Selection, Crossover, and Mutation is repeated.
- The goal is for each new generation to have a higher average fitness than the last one.
- The process continues until a stopping condition is met, such as a maximum number of generations or a sufficiently good solution being found.
- GA is great for problems where the solution space is vast and complex.

### Flowchart

`Akhil add flowchart`

---

## Differential Evolution

### What is it anyway?

- **Creators**: Kenneth Price and Rainer Storn in the 1990s.
- **Inspiration**: Like GA, it's an evolutionary algorithm inspired by biological evolution, but it's simpler and has a different approach.
- **Key Idea**: It uses vector differences to explore the search space, which is where the "differential" part of the name comes from.
- The algorithm operates on a population of real-valued vectors.
- Instead of using binary strings and crossover like GA, it relies on vector subtraction and addition.

### The Core Operators

- DE has three main phases: mutation, crossover, and selection.
- **Mutation**: For each individual vector in the population, a "mutant vector" is created. This is done by taking a base vector and adding the scaled difference of two other randomly selected vectors.
- The classic mutation is often expressed as:

$$v_i = x_r1 + F(x_r2 - x_r3)$$

- Here, $x_r1$, $x_r2$, $x_r3$ are three different vectors randomly chosen from the population, and F is a scaling factor.
- *Crossover*: The mutant vector is then combined with the original target vector to create a "trial vector."
- This is done element by element with a certain probability, called the crossover rate (CR). This ensures that the trial vector inherits some properties from both the original and the mutated vector.

## Selection and The Loop

- **Selection**: The final step is a greedy selection. The new trial vector is compared to the original target vector.
- If the trial vector has a better fitness value, it replaces the target vector in the population for the next generation; otherwise, the original vector is kept.
- The process repeats, with new individuals being generated, evaluated, and selected in each generation.
- Because DE uses a direct, vector-based approach, it's often simpler to implement and can converge faster than GA on certain types of problems.

## Flowchart

`Akhil add flowchart`

---