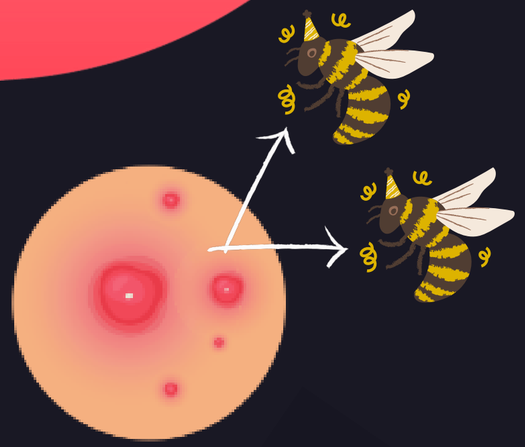
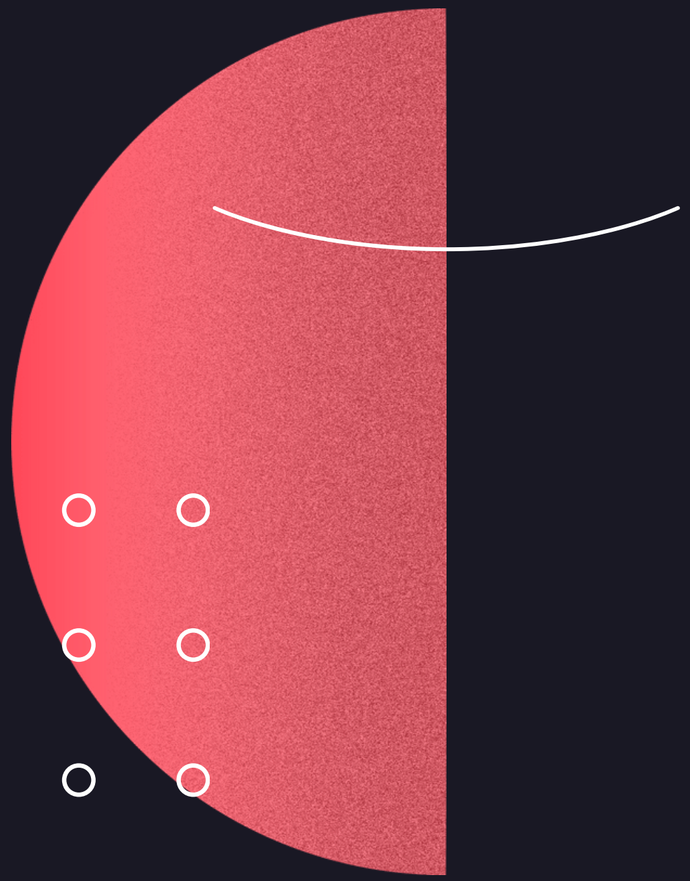


Skin Lesion Segmentation Method for Dermoscopy Images Using Artificial Bee Colony Algorithm

Team5

Team Members



Bhuvan.G

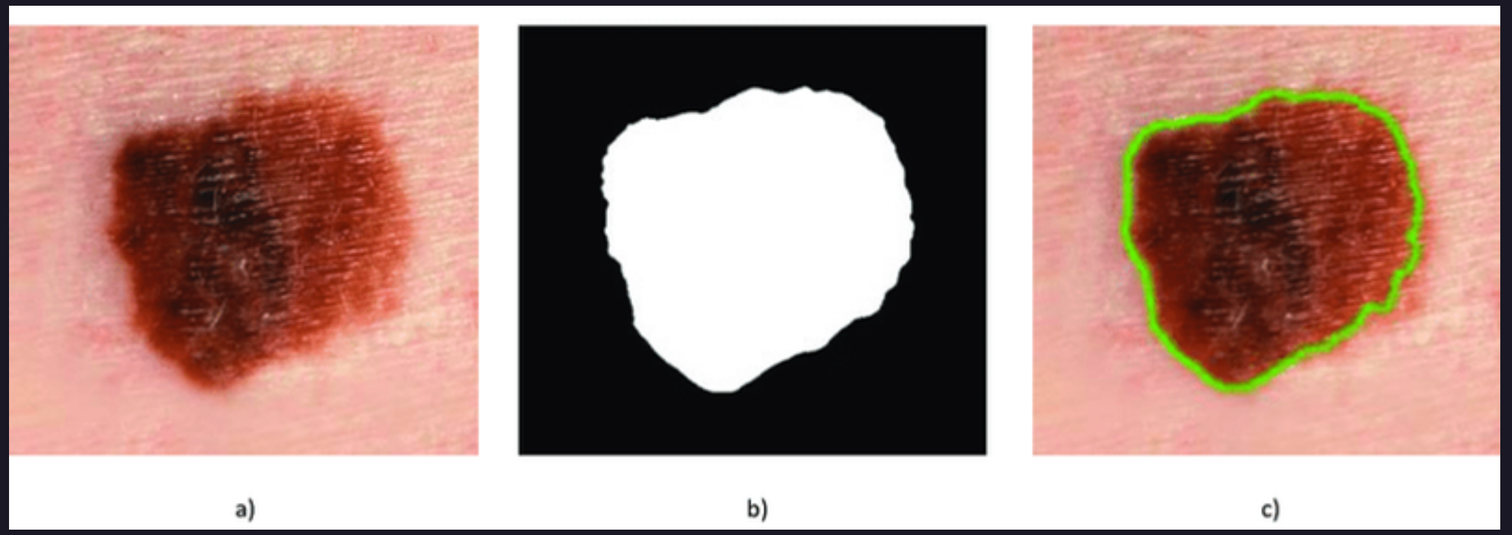
Lokesh Kumar.M

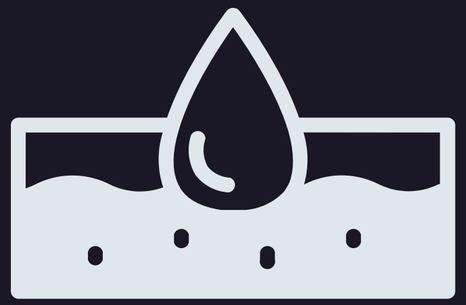
Sai Raghavi.T

Surya Venkata Teja.G

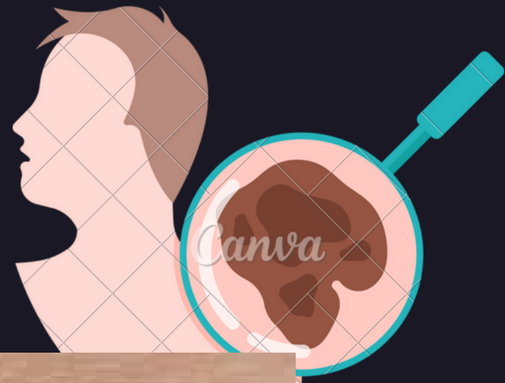
What is Skin Lesion Segmentation

Skin lesion segmentation, which is one of the medical image segmentation areas, is important for the detection of melanoma.





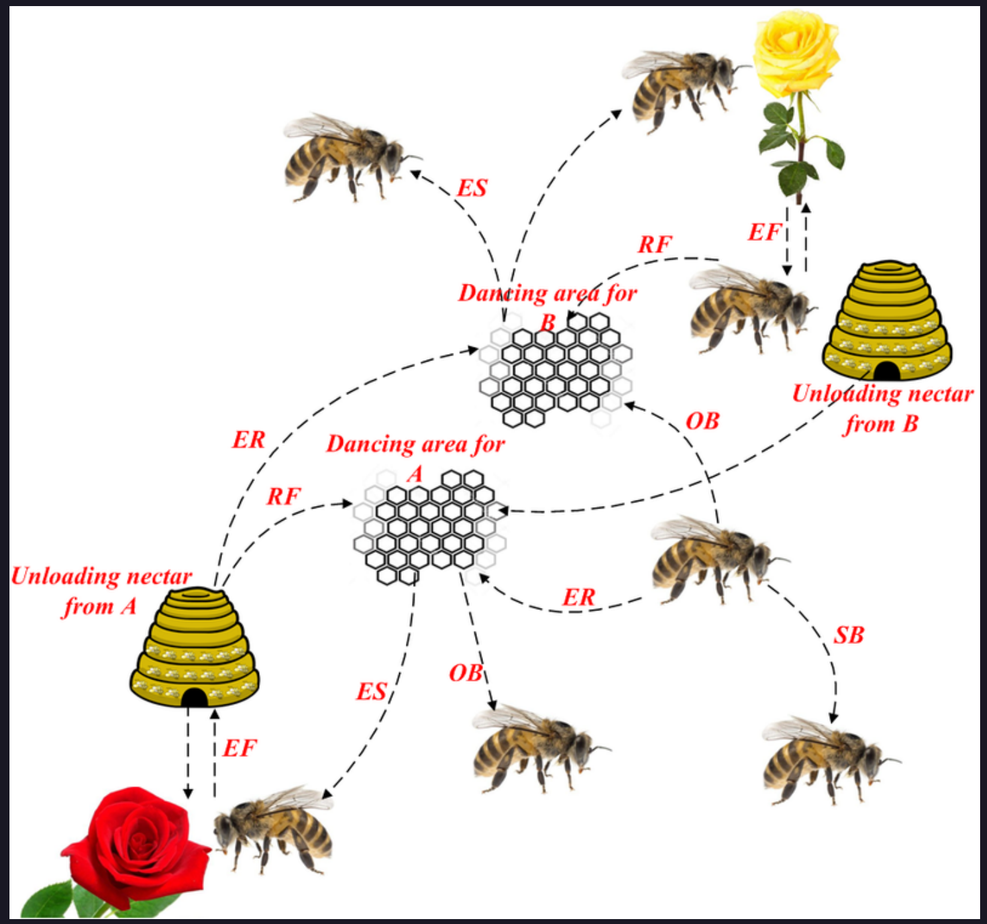
Why skin lesion segmentation is important in dermatology?



- Early Detection of Skin Cancer
- Accurate Diagnosis
- Tracking Lesion Growth
- Treatment Planning
- Efficient Telemedicine

What is an Artificial Bee Colony Algorithm?

- ABC algorithm derives inspiration from the foraging behavior of honeybees. In this process, bees search for food sources and convey their discoveries to other members of the hive. In the algorithm, the "food sources" are analogous to prospective resolutions to an optimization problem.



Steps followed by Artificial bee colony in skin lesion

1. Initialization

Initialize a population of scout bees, which represents potential solutions. Each bee corresponds to a possible set of features or parameters for detecting skin lesions.

2 Objective Function

Define an objective function, which measures the quality of a solution. In the context of skin lesion detection, this function could include measures like sensitivity, specificity, or accuracy of the detection system.

3. Employed Bees Phase

Select half of the bees to be employed bees.

Employed bees explore the search space and perform a local search around their current solutions.

Calculate the fitness of each employed bee's solution using the objective function:

$$Fitness_i = ObjectiveFunction(solution_i)$$

4. Onlooker Bees Phase

Select the other half of the bees to be onlooker bees.

Onlooker bees choose a solution to investigate based on the fitness of employed bees' solutions.

Calculate the probability of selecting each employed bee's solution:

$$P_i = Fitness_i / \sum(Fitness_i)$$

Choose an employed bee to follow based on the calculated probabilities.

Perform a local search around the selected solution.

5. Scout Bees Phase

If an employed bee's solution cannot be improved after a certain number of iterations, it becomes a scout bee.
Scout bees randomly generate new solutions to explore uncharted areas of the search space.

6. Memorize the Best Solution

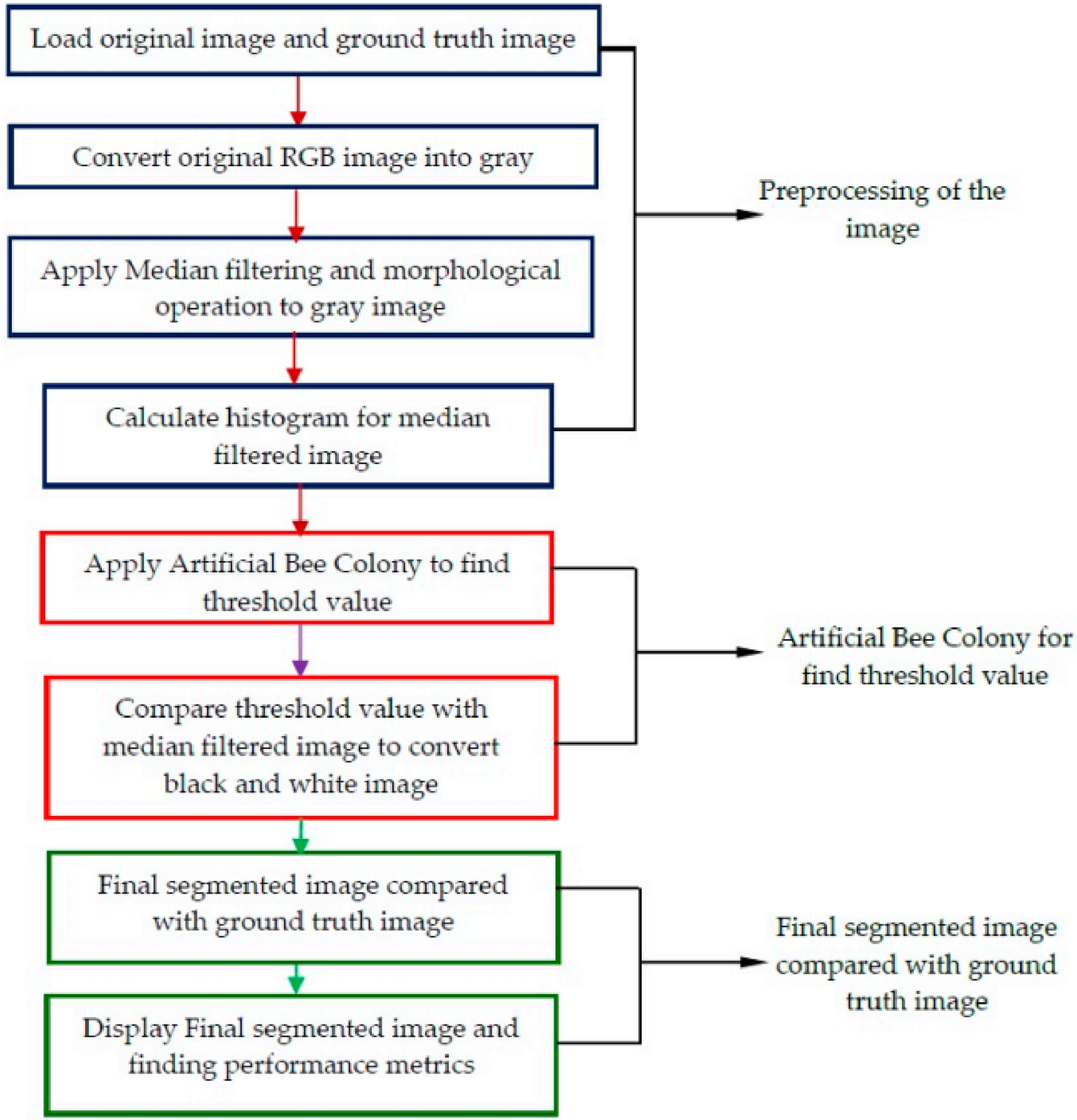
Keep track of the best solution found so far during the algorithm's execution.

7. Termination Condition

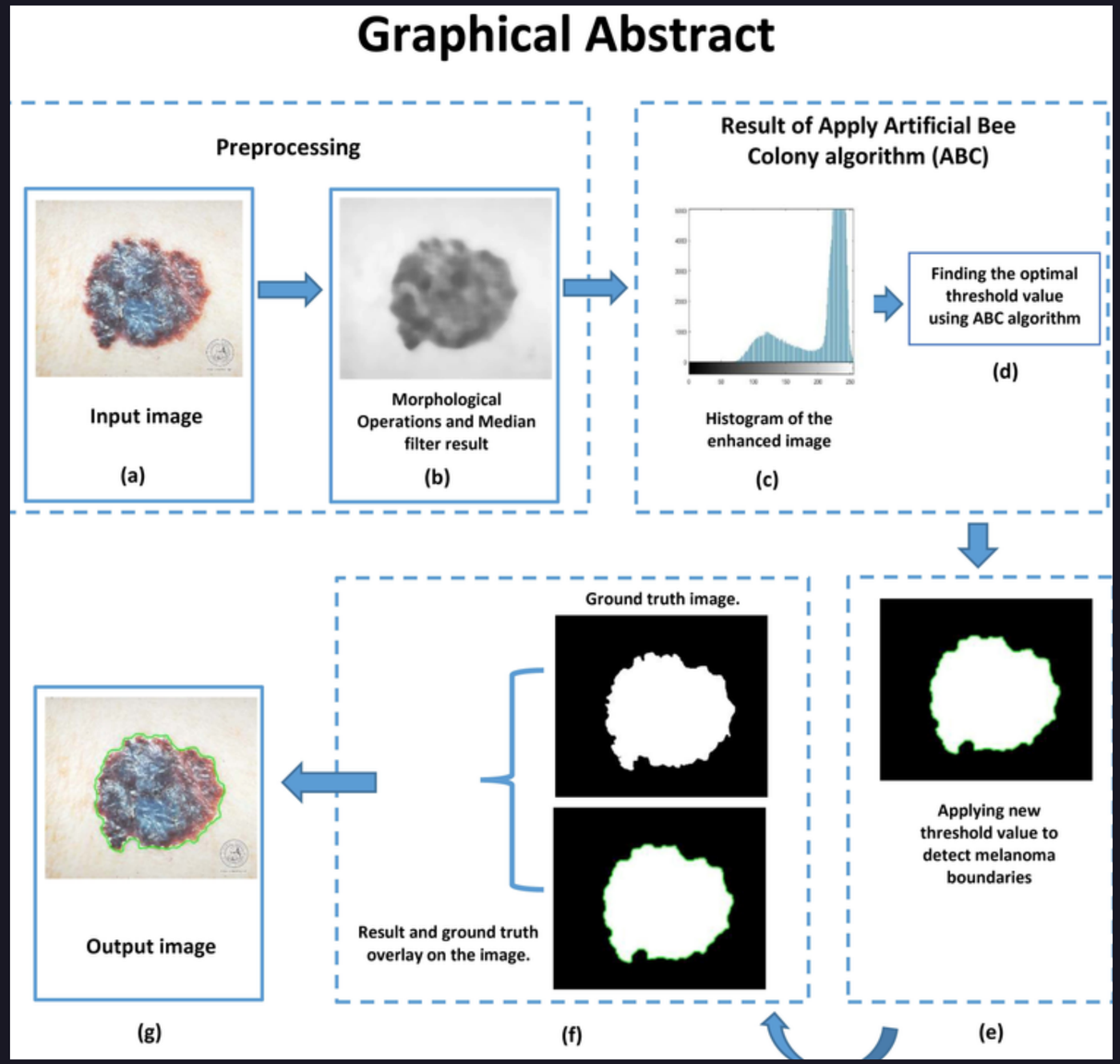
Determine a termination condition, such as a maximum number of iterations 8.Repeat or End

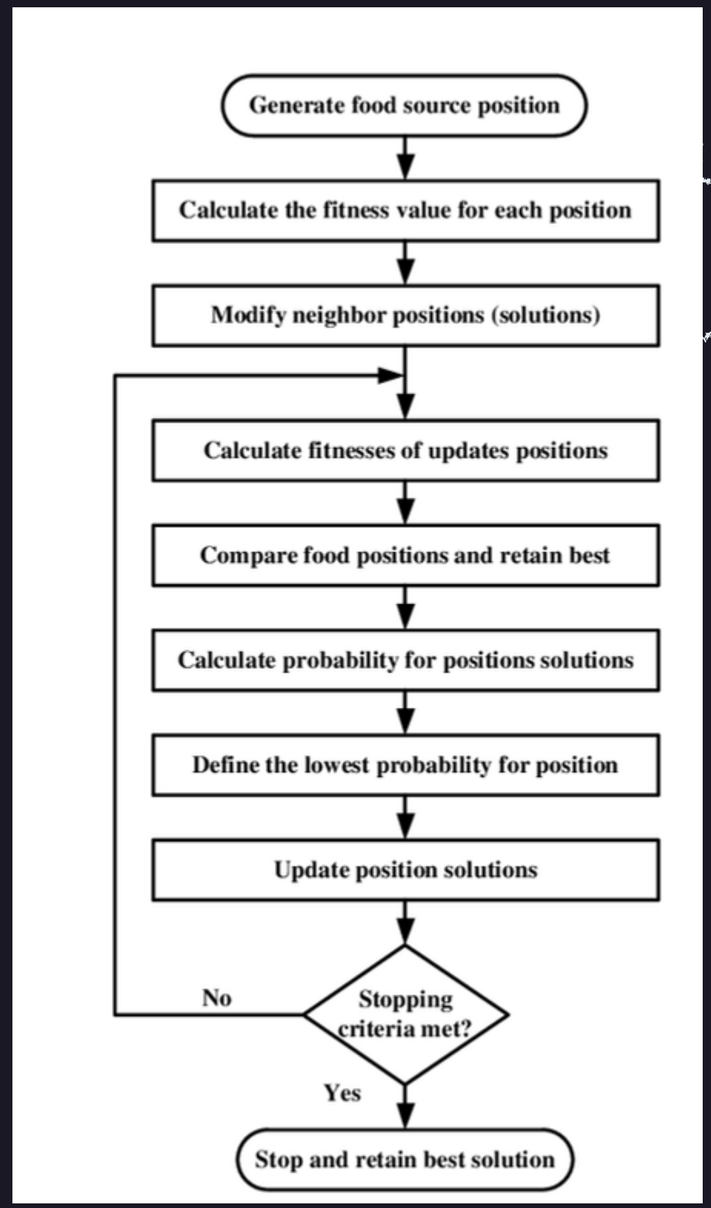
If the termination condition is not met, go back to Step 3.
If the termination condition is met, the algorithm terminates, and you can use the best solution found to detect skin lesions.

Flowchart for artificial bee colony-based melanoma detection



How is the Artificial Bee Colony Algorithm used for detecting Skin Disease?

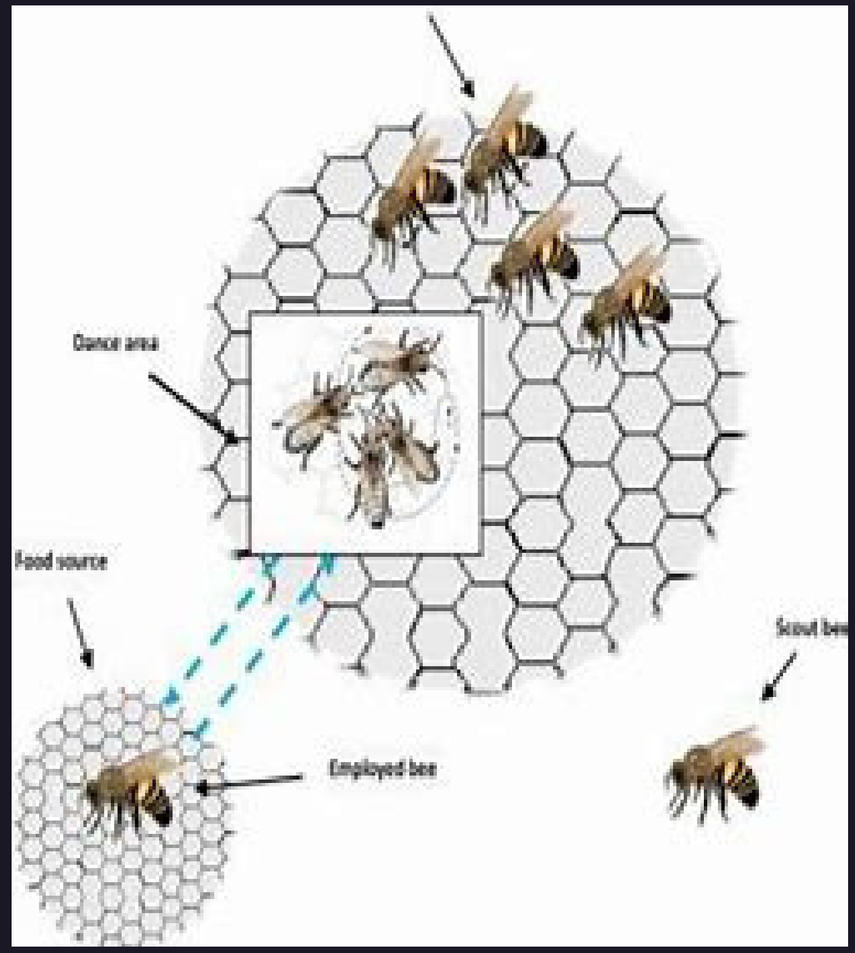




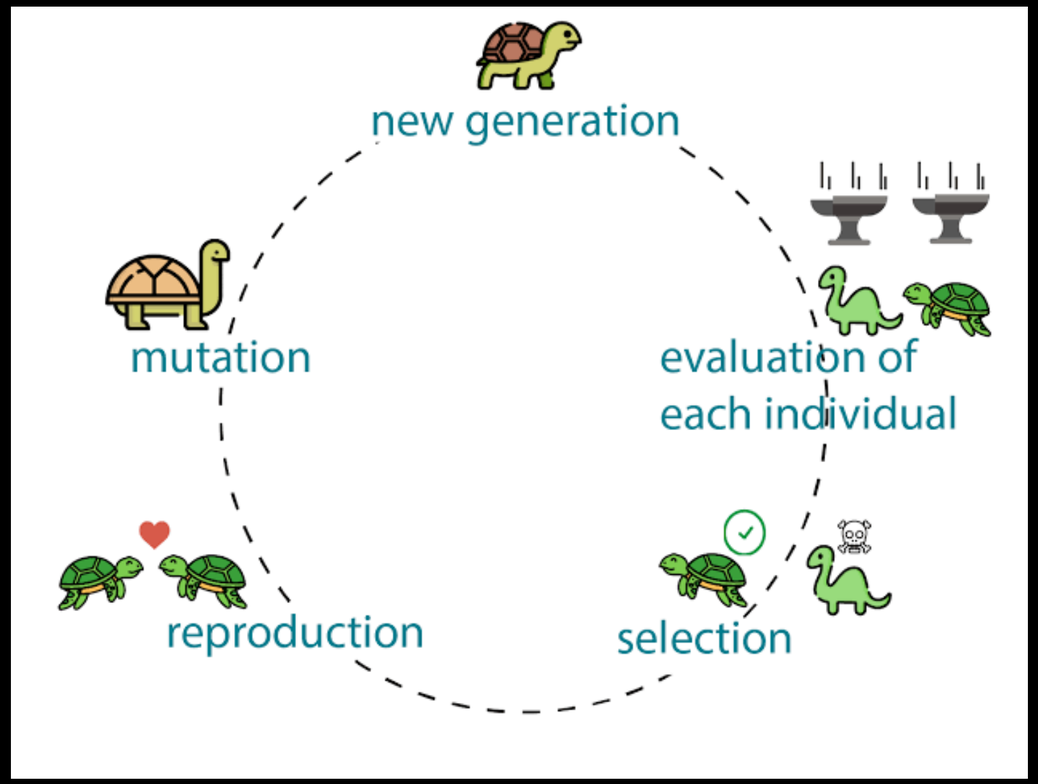
Applications

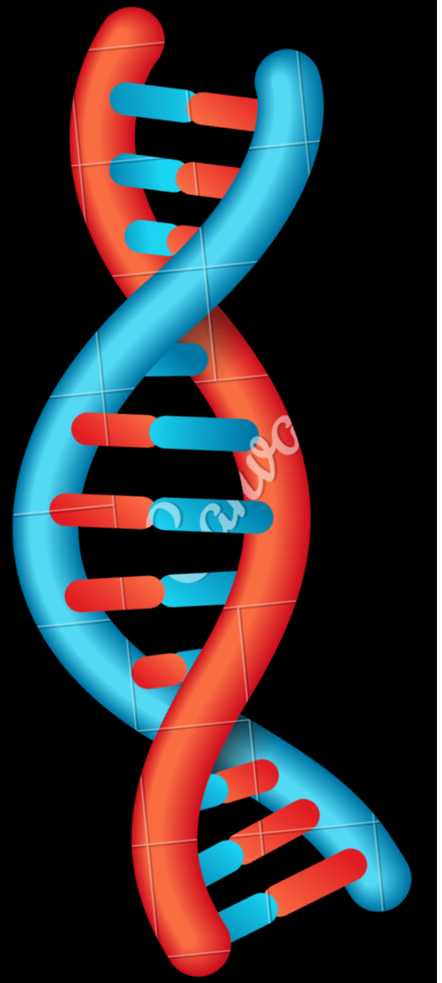
- One potential application of skin lesion segmentation using the artificial bee colony algorithm is in the field of dermatology

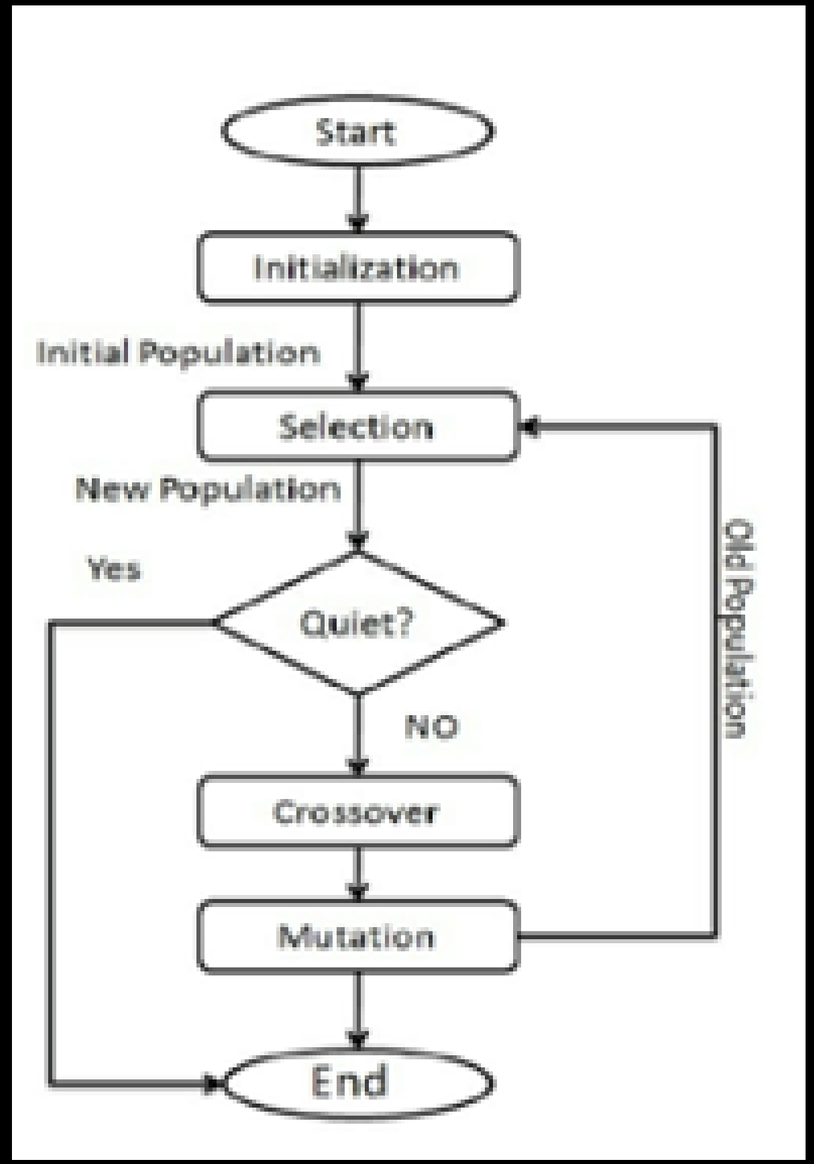
- The artificial bee colony algorithm can be used to monitor changes in skin.



Genetic algorithm







Steps followed by Genetic algorithm in skin lesion

Initialisation:

Population Initialization: Generate an initial population of chromosomes, each representing a set of parameters or thresholds for image segmentation.

"N_pop" be the population size

"N_genes" is the number of genes in each chromosome.

```
Population: [Chrom_1, Chrom_2, ..., Chrom_N_pop],
where each Chrom_i = [Gene_1, Gene_2, ..., Gene_N_genes]
```

Selection:

Fitness Evaluation: Calculate the fitness of each chromosome based on an objective function that quantifies the segmentation quality.

```
Fitness (Chrom_i) = F(Chrom_i), where "F" is the
fitness function.
```

Select Parents: Use a selection method to choose parent chromosomes for reproduction. The probability of selecting a chromosome "Chrom_i" can be defined as

```
P_select(Chrom_i) = Fitness(Chrom_i) / Σ(Fitness(Chrom_j) for all chromosomes)
```

Crossover :

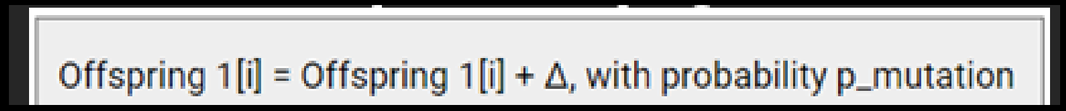
Select pairs of parent chromosomes, e.g., "Parent1" and "Parent2."

Perform crossover at a specific crossover point "k," where 1 <= k <= N_genes, to create offspring chromosomes

```
Offspring1 = Parent1[1:k] + Parent2[k+1:N_genes]
Offspring2 = Parent2[1:k] + Parent1[k+1:N_genes]
```

Mutation:

Introduce random mutations in some genes of the offspring chromosomes to maintain diversity. The mutation rate "p_mutation" is the probability that a gene mutates. For example, for "Offspring1"



Δ is a random mutation value

Termination Conditions:

Define termination conditions, such as a maximum number of generations (G_max), a target fitness value, or convergence conditions. The GA continues until these criteria are met.

Repeat Steps 2-6:

Continue the selection, crossover, mutation, and fitness evaluation process for multiple generations, incrementing the generation counter (G) at each iteration.

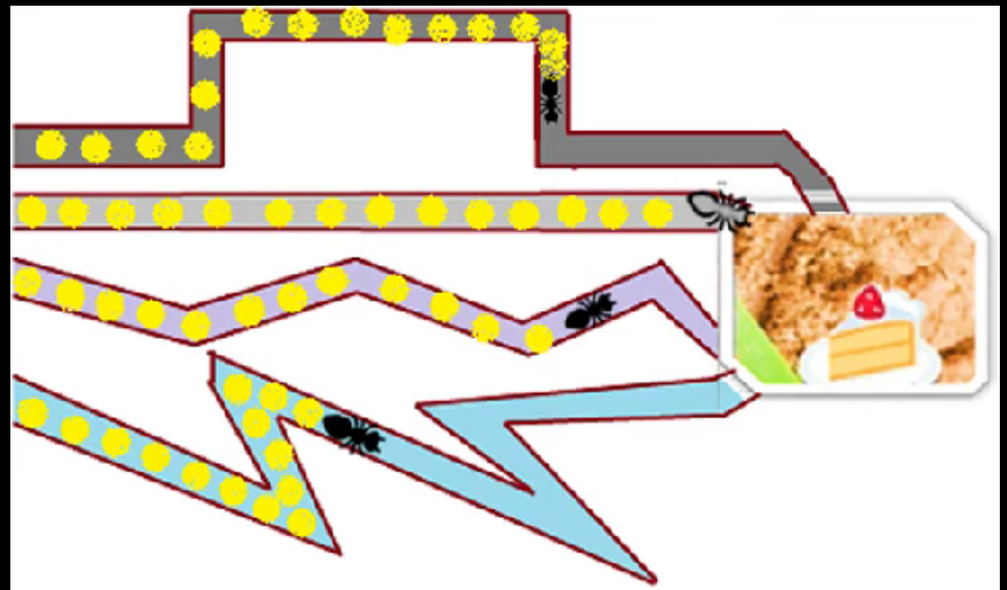
Final Result:

After the GA's execution, the chromosome with the best fitness represents the optimized set of parameters and thresholds for skin lesion segmentation

ANT COLONY OPTIMIZATION ALGORITHM

Inspired by ants' behavior in finding the shortest path between their colony and food sources, Ant Colony Optimization (ACO) is a metaheuristic algorithm. This technique is frequently used to address optimization problems in various fields, including computer science and engineering.

Ant Colony Optimization (ACO) algorithm is inspired by social behavior of real ants.
ACO is basically inspired by pheromone based Ant communication.
ACO is developed by Marco Dorigo in 1922.
This technique is used to find Optimal paths.



1. Initialize ACO parameters

PopulationSize(k)

MaxT

Tau

Alpha

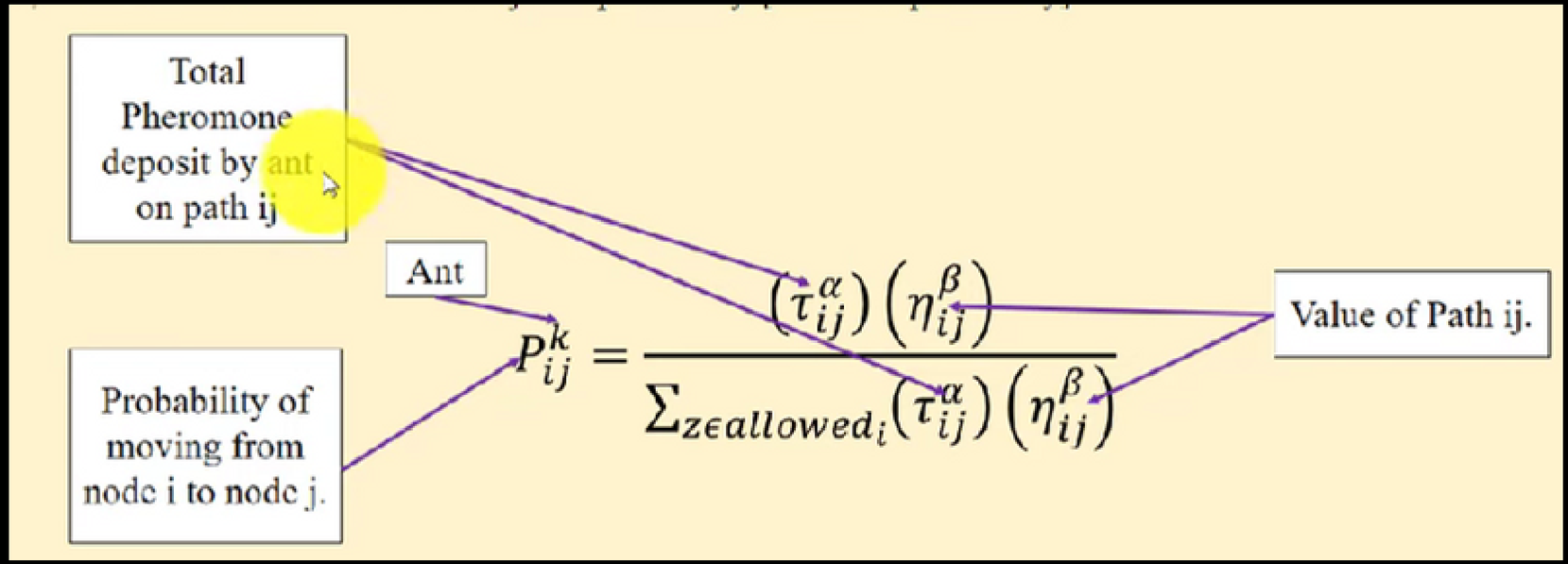
Beta

rho

2. Ant solution Construction:

Current Iteration = 1 to Max number of iterations

3. Position each ant in stating node :



4. Repeat until ant build the best solution, then Compute the fitness valuc.

5. Update best solution.

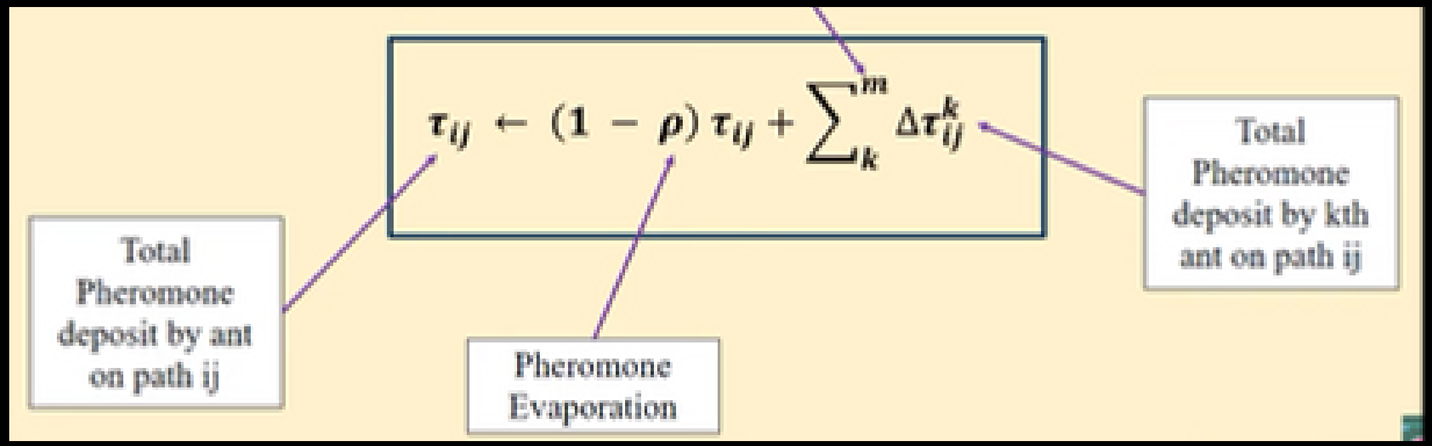
Compare the best solution with each ant solution.

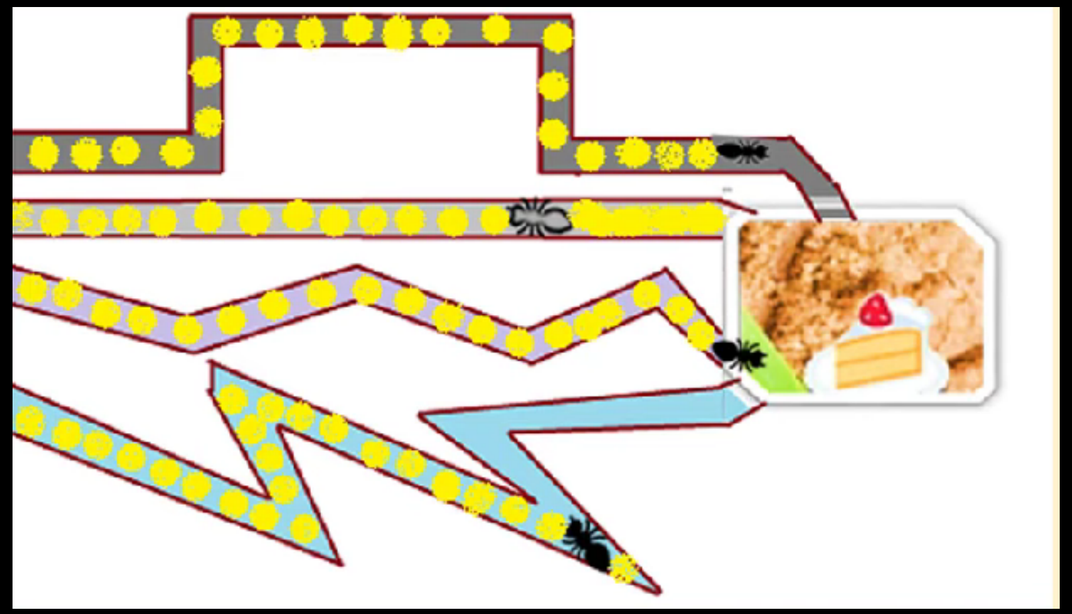
If (Ant(3) Solution < Best Solution)

Consider Ant(3) best solution.

Ant (k =PopulationSize)

8. pheromone update.

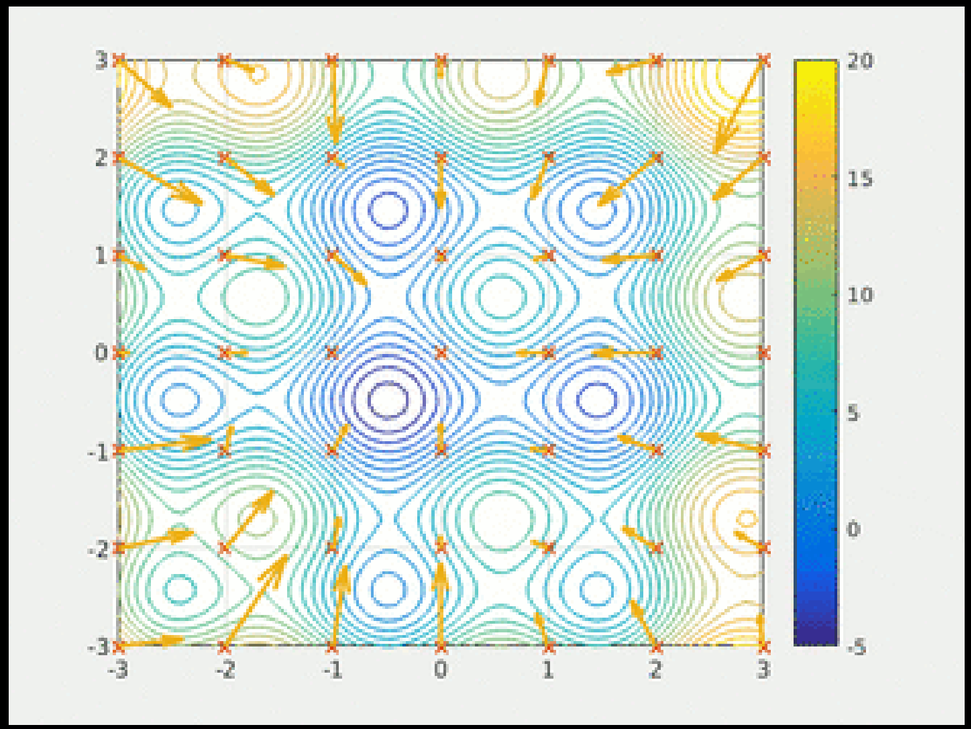




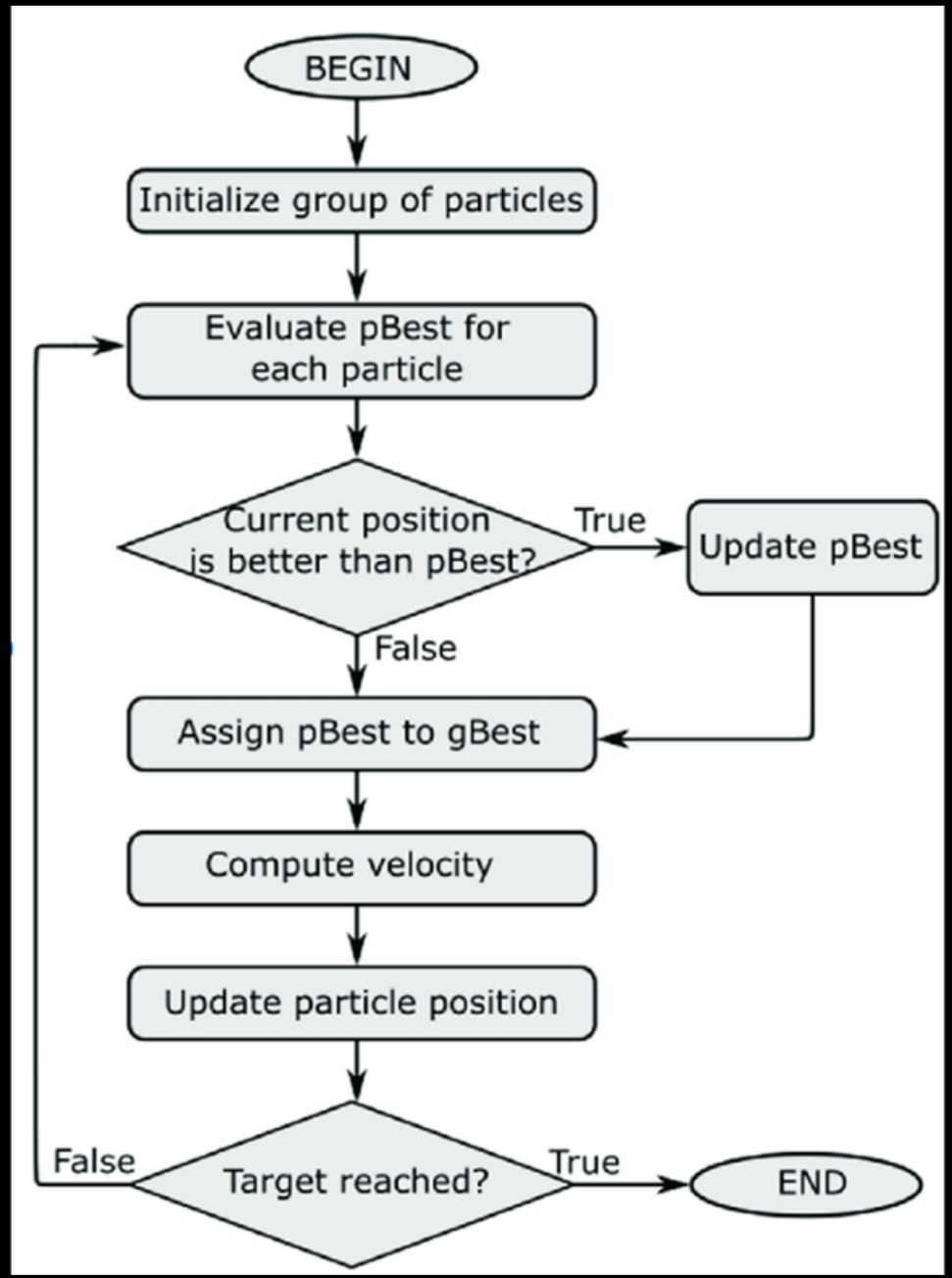
Particle Swarm Optimization (PSO)

Key Facts about Particle Swarm Optimization (PSO) technique:

- **PSO is an evolutionary and stochastic optimization technique inspired by nature, used to solve complex optimization problems.**
- **This robust technique relies on swarm movements and intelligence to achieve results.**
- **Developed by James Kennedy and Russ Eberhart in 1995, PSO has proven effective in solving a wide variety of optimization and search problems.**



Algorithm Workflow - Skin Lesion Segmentation Method using Particle Swarm Optimization (PSO) Algorithm.



1. Initialize a swarm of particles.

This involves generating a random population of particles, each of which represents a potential solution to the optimization problem. The particles are typically initialized within the bounds of the search space.

2. Evaluate the fitness of each particle.

The fitness of each particle is evaluated using the objective function of the optimization problem. The objective function is a mathematical expression that quantifies the quality of a solution.

3. Update the personal best position of each particle.

The personal best position of each particle is the position at which it achieved the best fitness value so far.

4. Update the global best position of the swarm.

The global best position of the swarm is the position of the particle with the best fitness value so far.

5. Update the velocity of each particle.

The velocity of each particle is updated based on its personal best position, the global best position and two random numbers.

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{Diversification}} + \underbrace{c_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t) + c_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{Intensification}}$$

6. Update the position of each particle.

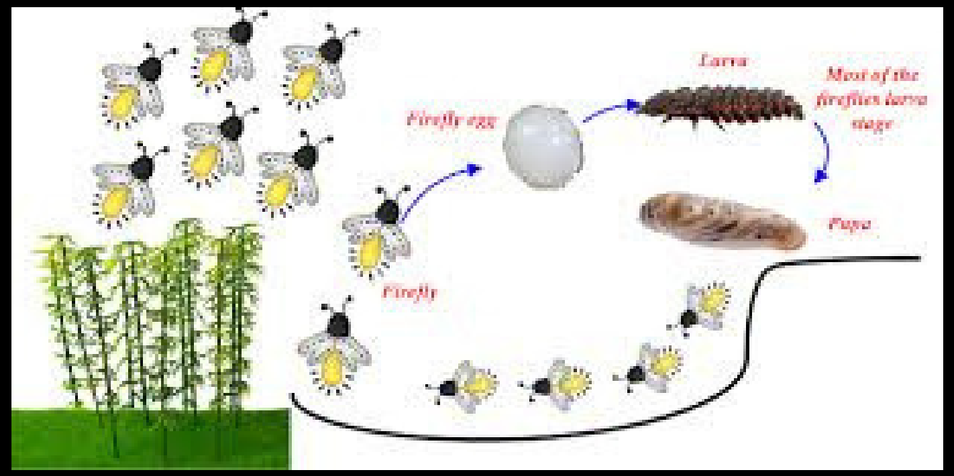
$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \mathbf{v}_i^{t+1}$$

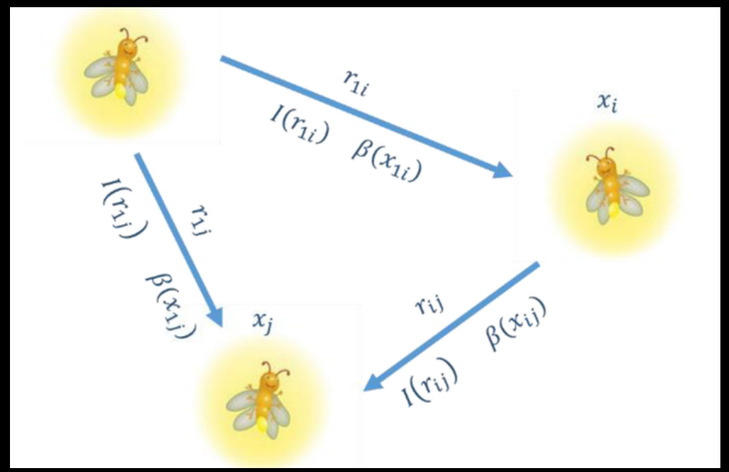
The position of each particle is updated based on its current velocity.

7. Repeat steps 2-6 until the termination criterion is met.

Firefly Algorithm

- The Firefly Algorithm is a nature-inspired optimization technique.
- Mimics the flashing behavior of fireflies.
- Used to find optimal solutions in optimization problems.
- Fireflies attract each other based on brightness (fitness).
- Brighter fireflies attract others and move closer.





Firefly Algorithm

Initialization:

1. Initialize a population of fireflies with random positions and set their initial light intensity (fitness).

Light Intensity Calculation:

2. Calculate the light intensity of each firefly based on the objective function of the optimization problem. A brighter firefly has a higher fitness value.

Movement Towards Brighter Fireflies:

3. Each firefly is attracted to brighter fireflies in the population. The attractiveness of one firefly to another is inversely proportional to their distance.

Firefly Attraction Formula:

$$\text{Attractiveness}(i, j) = \beta * \exp(-\gamma * \text{Distance}(i, j)^2)$$

Movement Towards Brighter Fireflies:

Fireflies move toward brighter neighbors by adjusting their positions based on the attractiveness values. The movement is guided by the formula:

NewPosition(i) = CurrentPosition(i) + StepSize * (Attractiveness(i, j) * (Firefly_j_Position - Firefly_i_Position)) + ε is a randomization factor

Update Light IntensityUpdate Light Intensity:

5. After moving, update the light intensity of each firefly based on the new positions and the objective function.

Iteration:

6. Repeat steps 2 to 5 for a specified number of iterations or until a termination condition is met.

Termination:

7. The algorithm terminates when the specified number of iterations is reached or when a stopping criterion, such as achieving a desired fitness threshold, is satisfied.

Algorithm	Strengths	Weaknesses	Best suited for
Artificial Bee Colony (ABC)	- Easy to implement	- Can get trapped in local optima	- Global optimization problems
Ant Colony Optimization (ACO)	- Robust to noise	- Can be slow to converge	- Combinatorial optimization problems
Particle Swarm Optimization (PSO)	- Fast convergence	- Can get trapped in local optima	- Continuous optimization problems
Genetic Algorithm (GA)	- Flexible and powerful	- Can be slow to converge	- Global optimization problems
Firefly Algorithm (FA)	- Good at escaping local optima	- Can be sensitive to parameter settings	- Continuous optimization problems

Thank You