COURSE CODE: 19CSE422A

COURSE TITLE: COMPUTATIONAL INTELLIGENCE

UNIT 4 – SWARM INTELLIGENCE

Dr. Monica S Ravishankar

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Ramaiah University of Applied Sciences

AGENDA

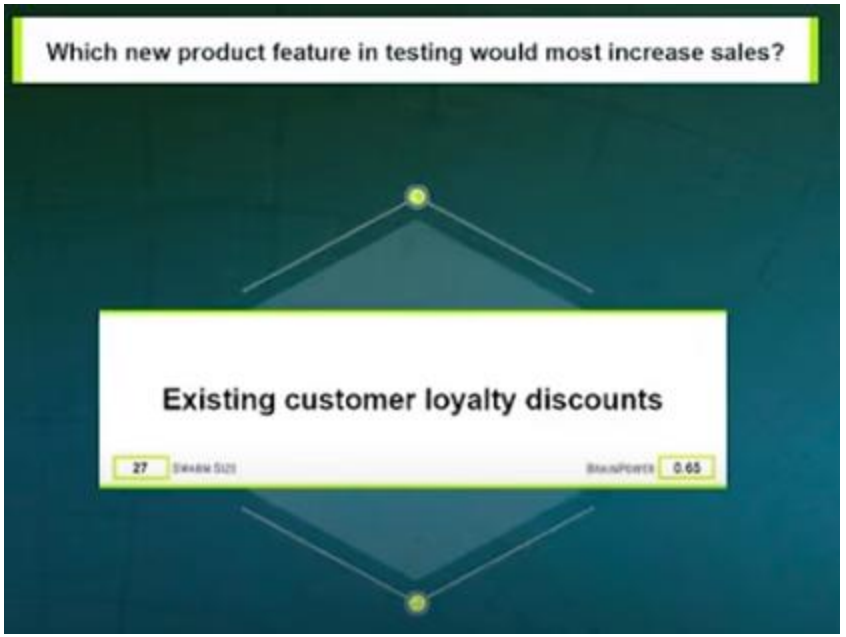
- Introduction to Swarm Intelligence
- Particle Swarm Optimization Algorithm
- Bacterial Foraging Algorithm
- Artificial Honey Bee
- Ant Colony Optimization
- Recent Trends in SI
- Variants and Hybrids of SI

SWARM INTELLIGENCE

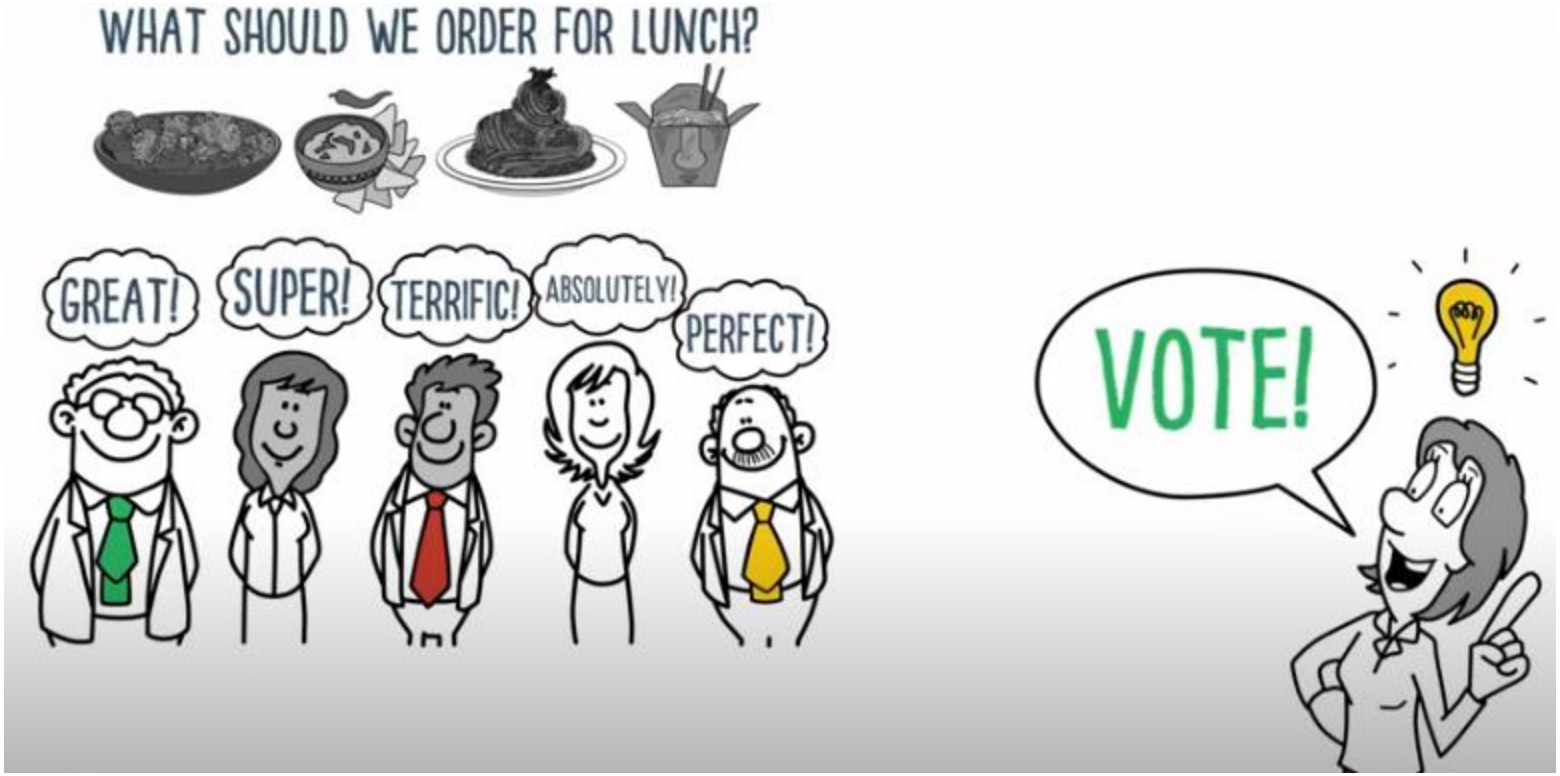
SWARM INTELLIGENCE

SWARM INTELLIGENCE

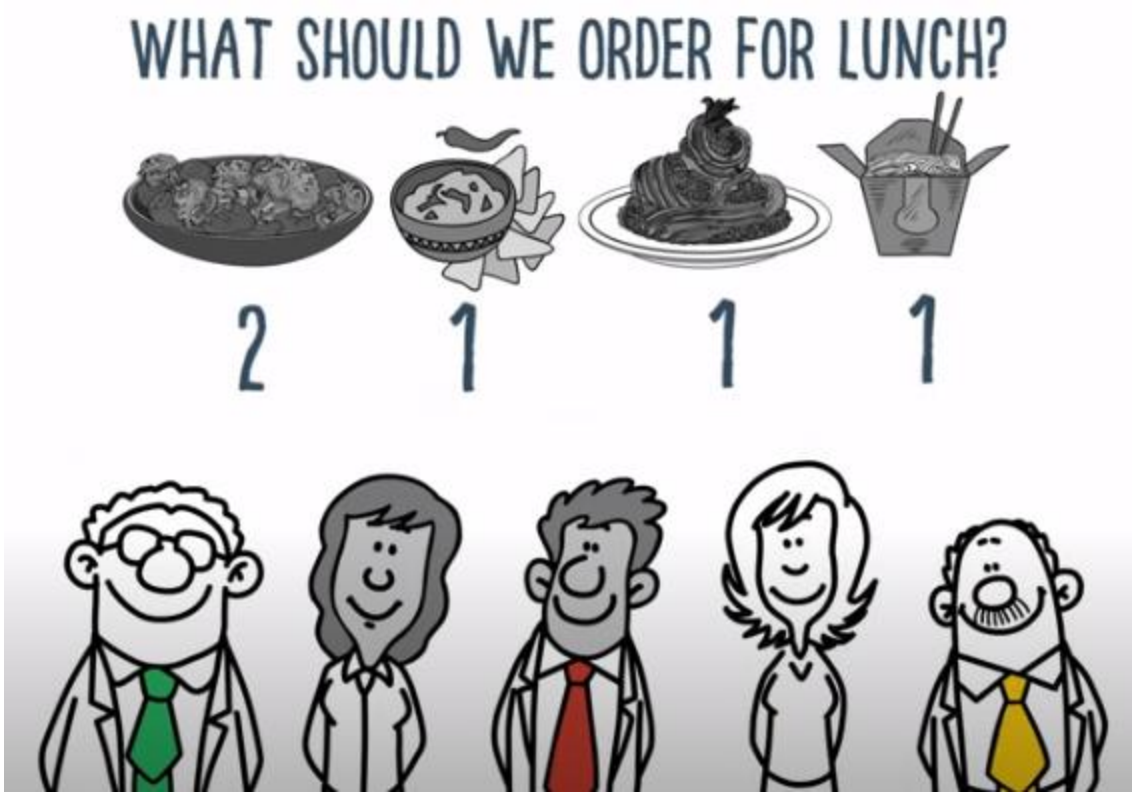
SWARM INTELLIGENCE



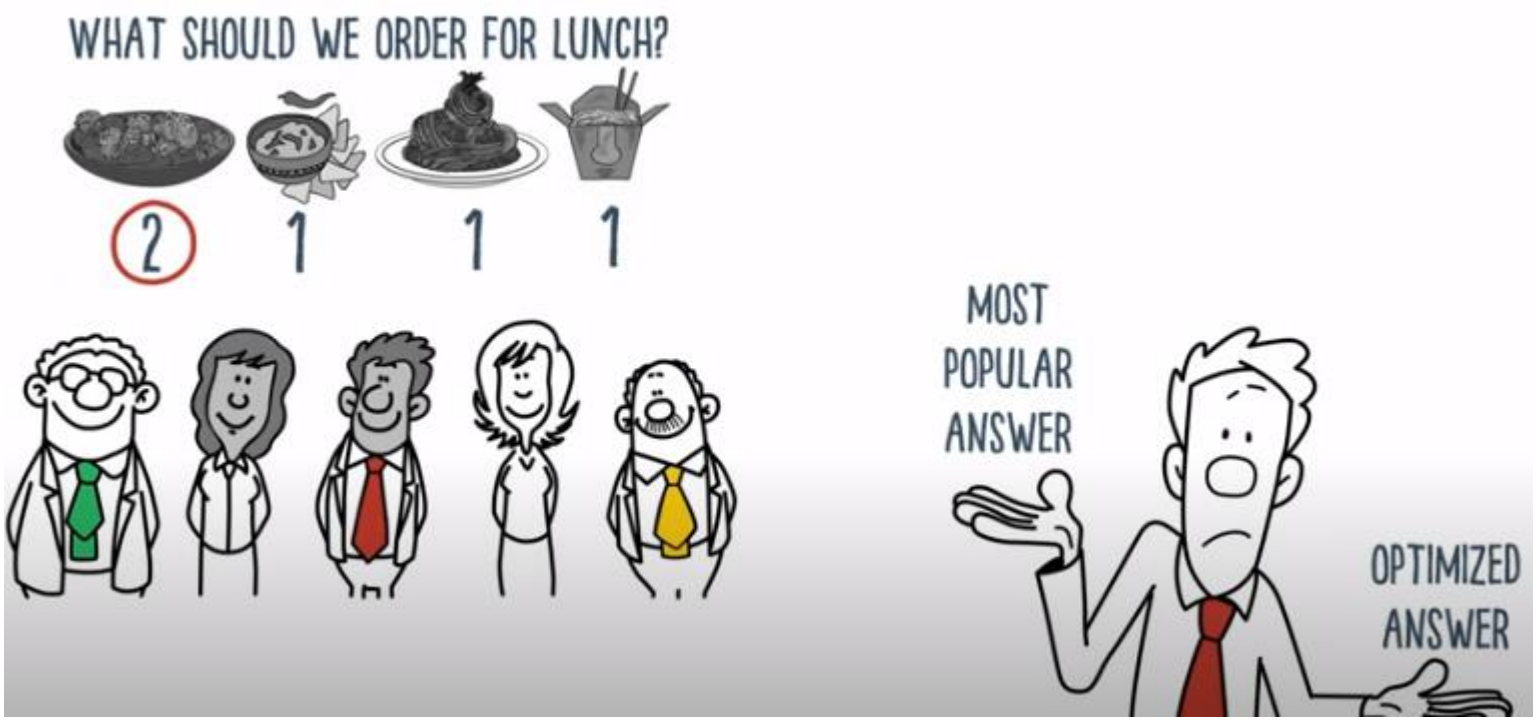
SWARM INTELLIGENCE



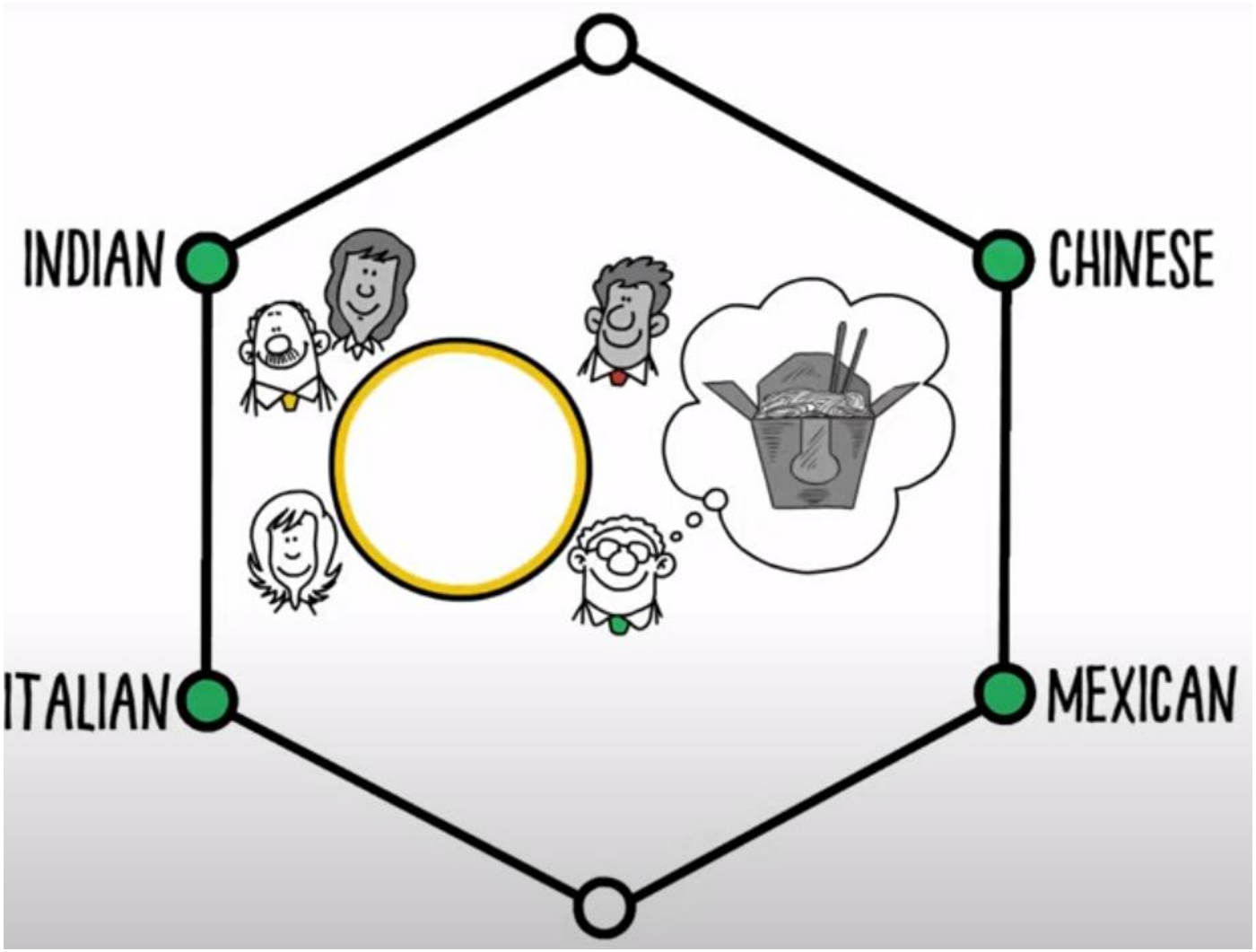
SWARM INTELLIGENCE



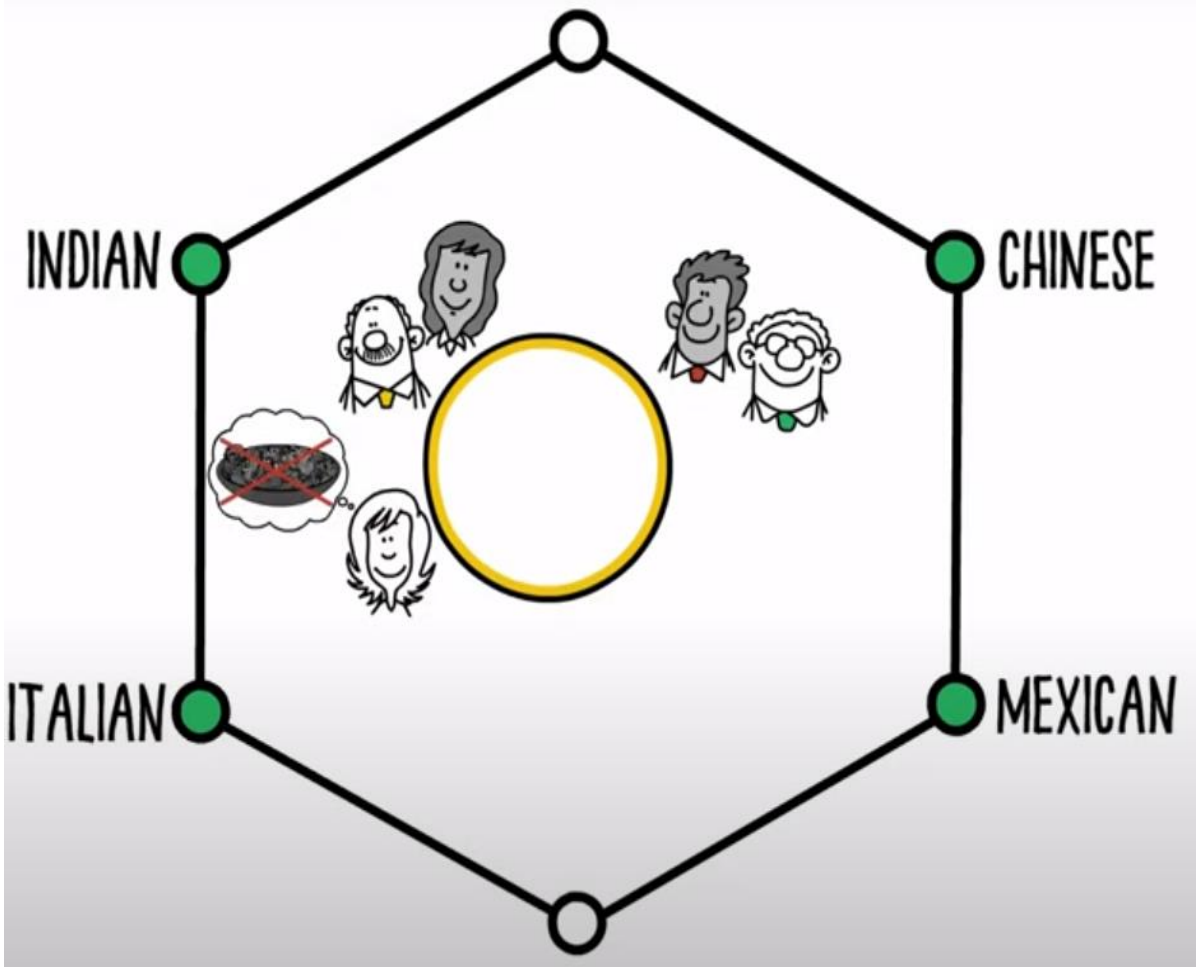
SWARM INTELLIGENCE



SWARM INTELLIGENCE

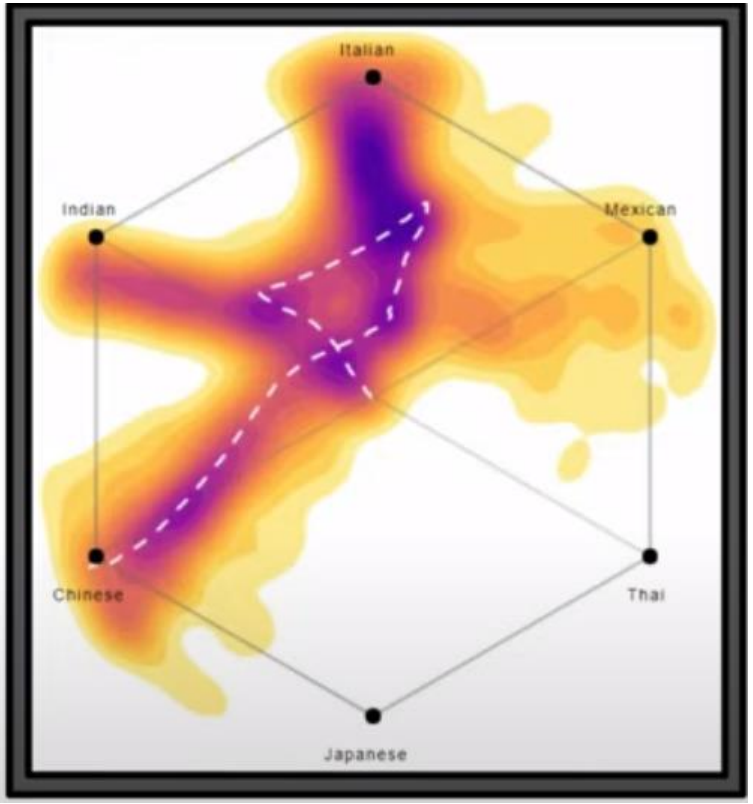


SWARM INTELLIGENCE

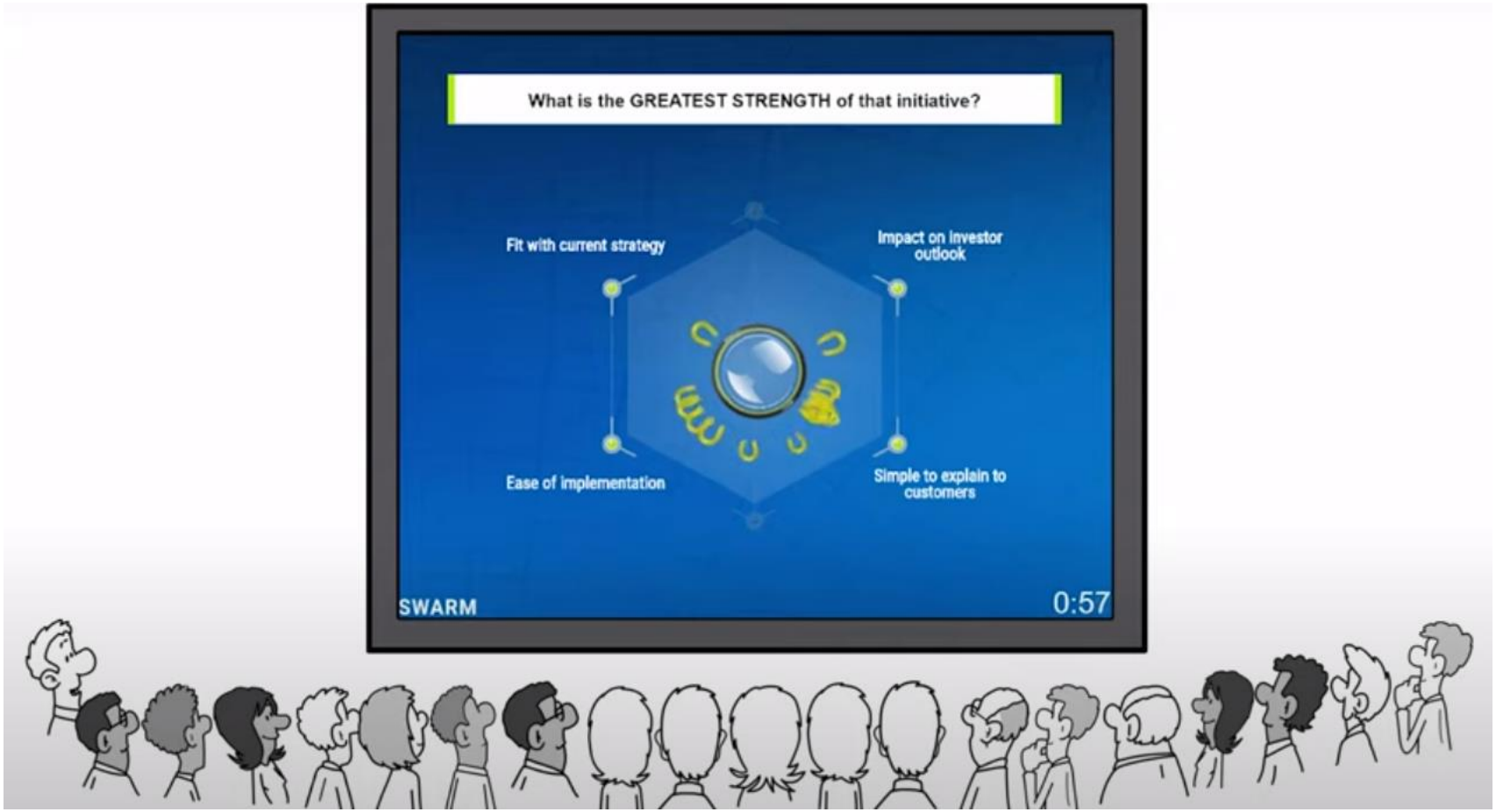


SWARM INTELLIGENCE

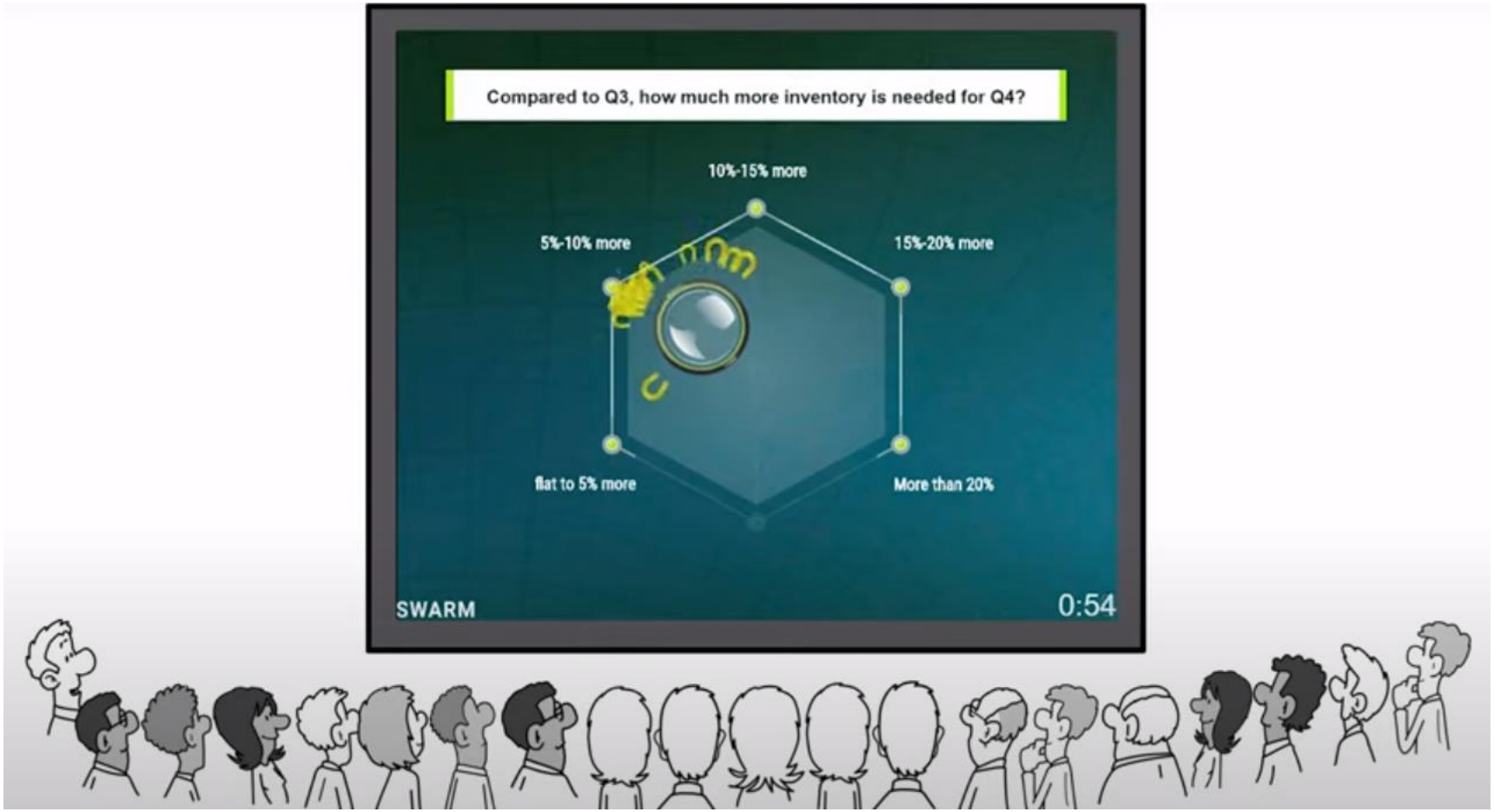
SWARM INTELLIGENCE



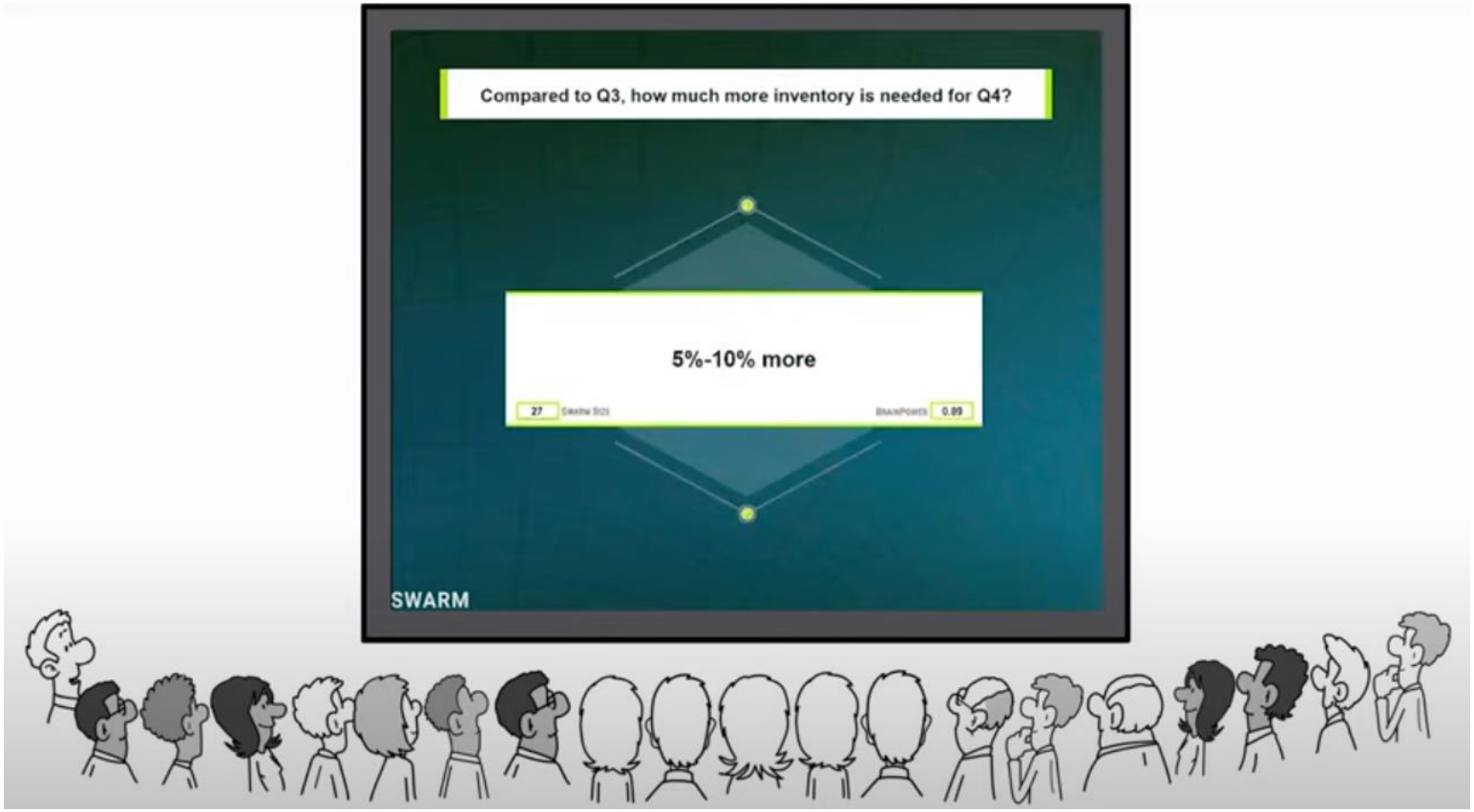
SWARM INTELLIGENCE



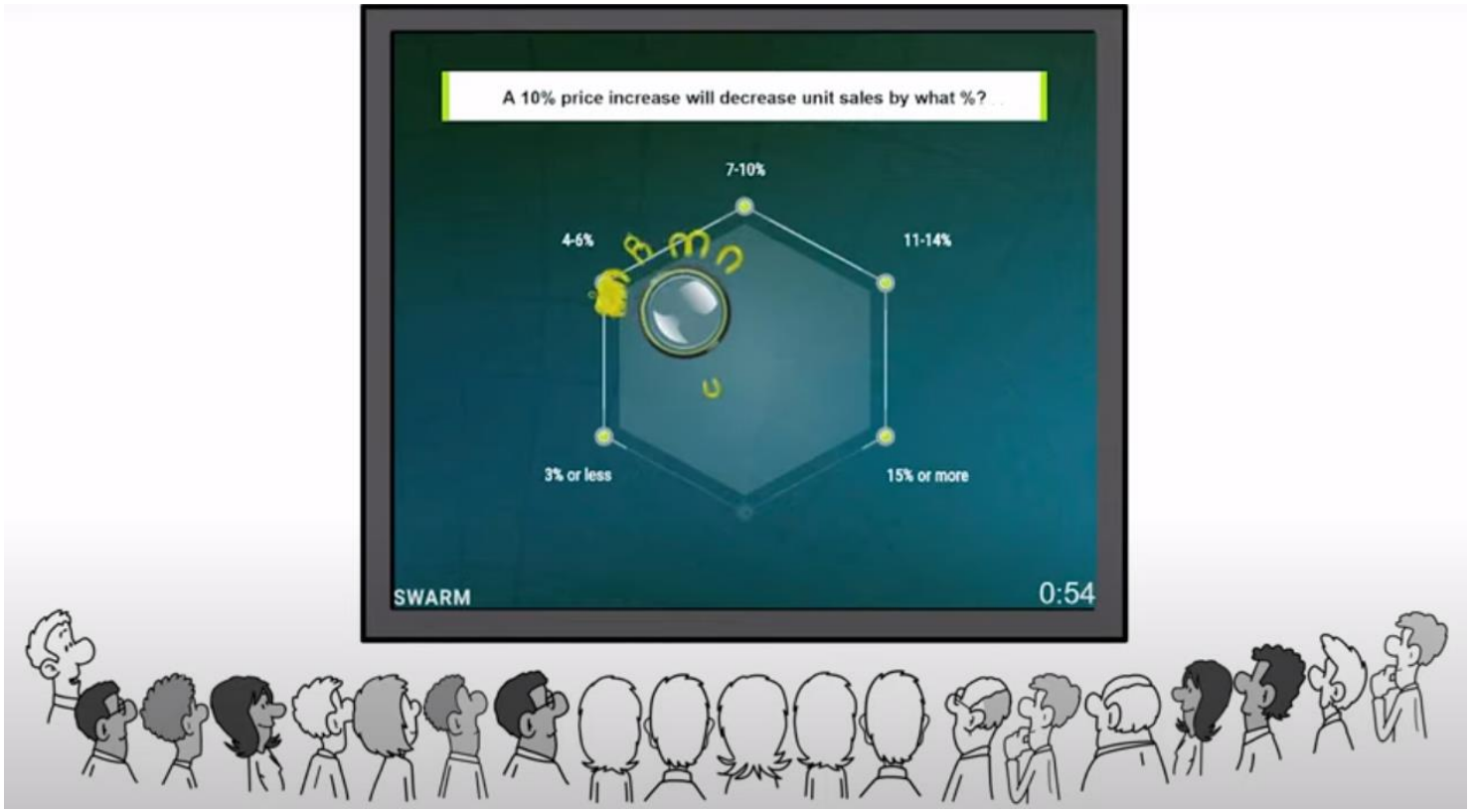
SWARM INTELLIGENCE



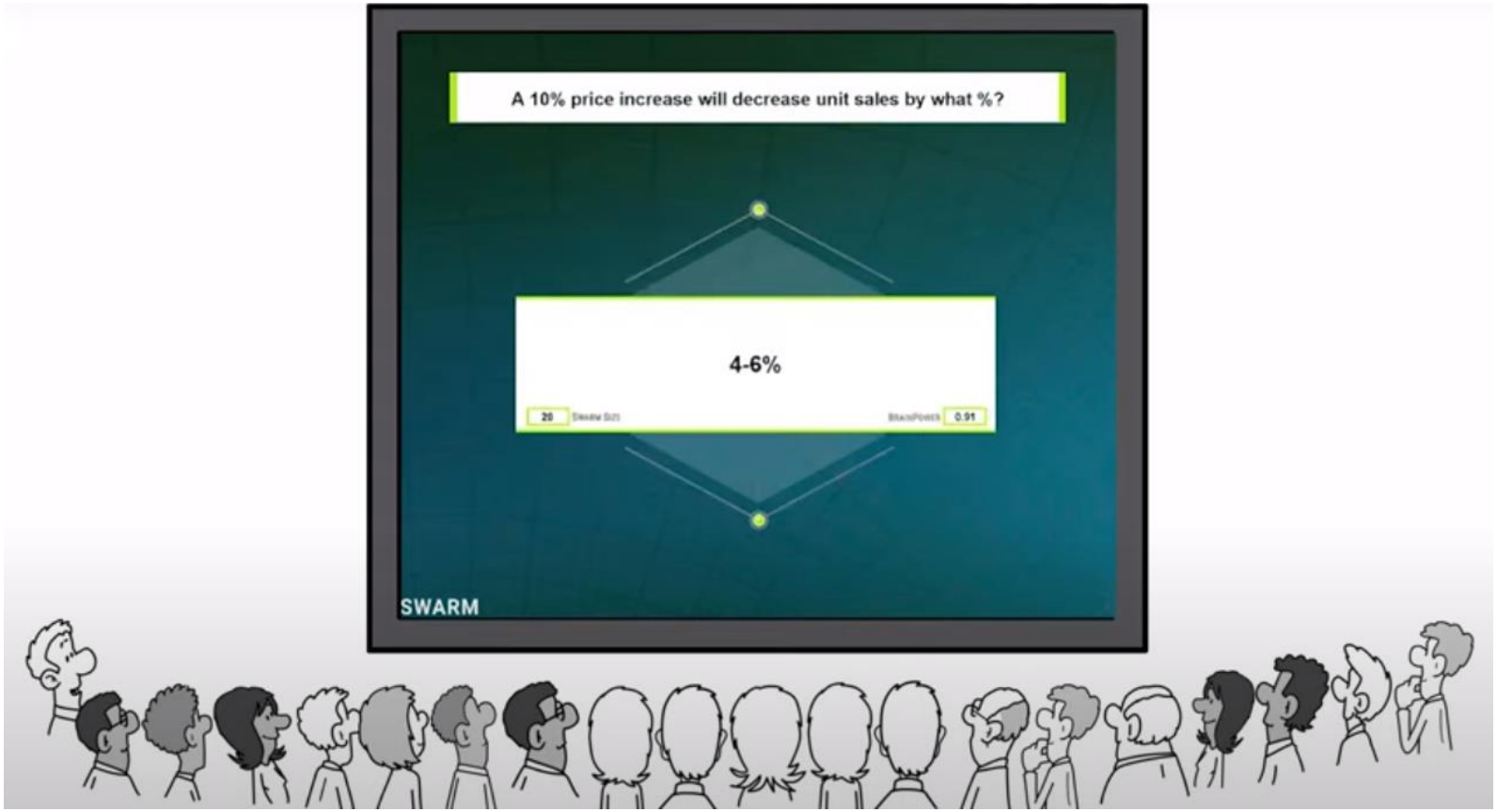
SWARM INTELLIGENCE



SWARM INTELLIGENCE



SWARM INTELLIGENCE



SI - DEFINITION

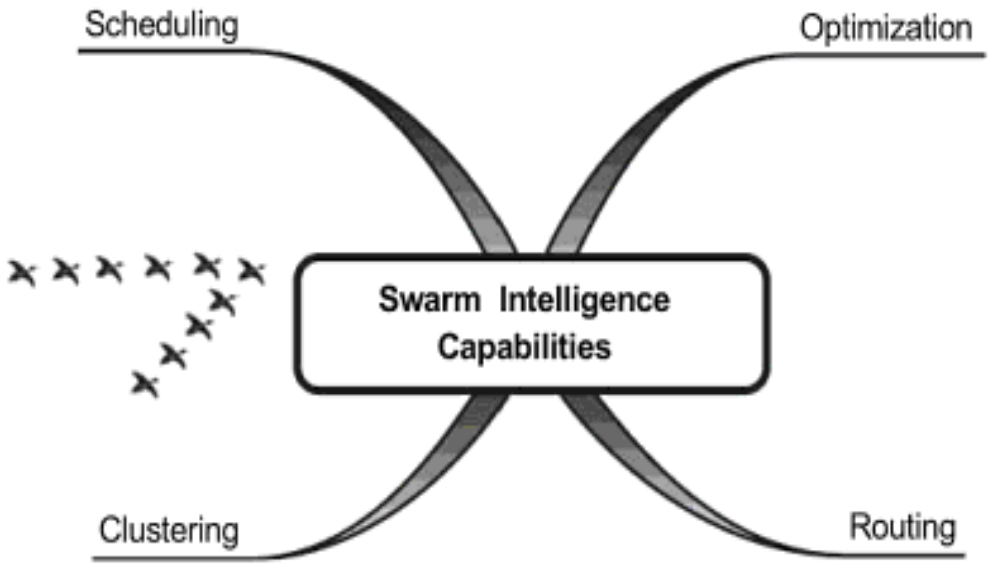
- Discipline that deals with natural and artificial systems composed of many individuals that coordinate using decentralized control and self-organization.

- Examples of systems
  - colonies of ants and termites,
  - schools of fish,
  - flocks of birds,
  - herds of land animals.
  - human artifacts - multi-robot systems, computer programs that are written to tackle optimization and
  - data analysis problems.

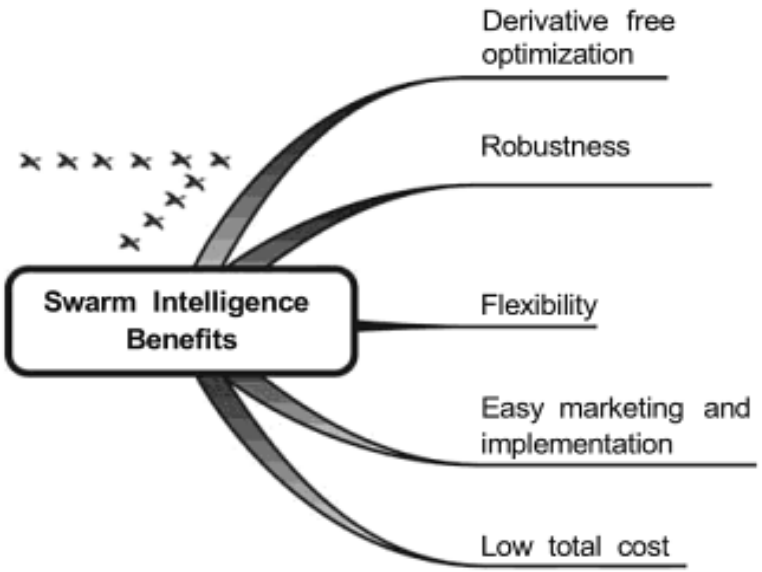
Properties of a Swarm Intelligence System

- It is composed of many individuals;
- The individuals are relatively homogeneous;
- Interactions among the individuals are based on simple behavioral rules that exploit only local information that the individuals exchange directly or via the environment (stigmergy);
- The overall behavior of the system results from the interactions of individuals with each other and with their environment, that is, the group behavior self-organizes.

KEY CAPABILITIES OF SWARM INTELLIGENCE



Advantages of swarm intelligence



Swarm Principles

General Swarm Principles

1) Proximity principle

2) Quality principle

3) Principle of diverse response

4) Principle of stability

5) Principle of adaptability

SWARM TECHNOLOGIES

Particle Swarm Optimization PSO Search strategy

- **Inspiration:** Inspired by the social foraging behavior of some animals
- **Strategy:** To have all the particles locate the optima in a multi-dimensional space
- **PSO Rules**

1) Separation

2) Alignment

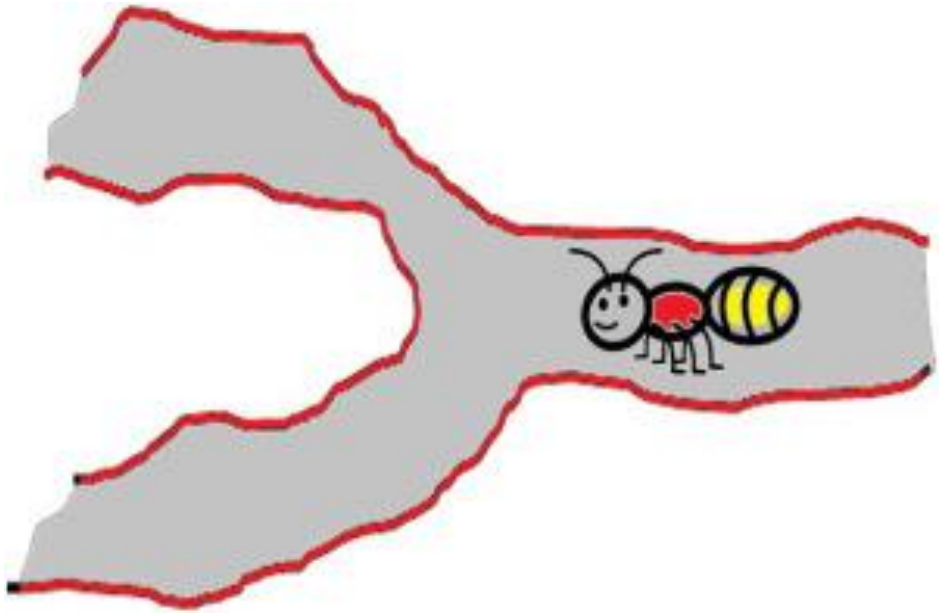
3) Cohesion

4) Desire factor per bird for roosting areas

SWARM TECHNOLOGIES

Ant System:

- **Inspiration:** It is inspired by the pheromone communication of the blind ants regarding a good path between colony and the food source in an environment
- **Strategy:** The objective of the strategy is to exploit historic



Ant Based Clustering Algorithm: To find the shortest way between the data items of a given data-set to be clustered.

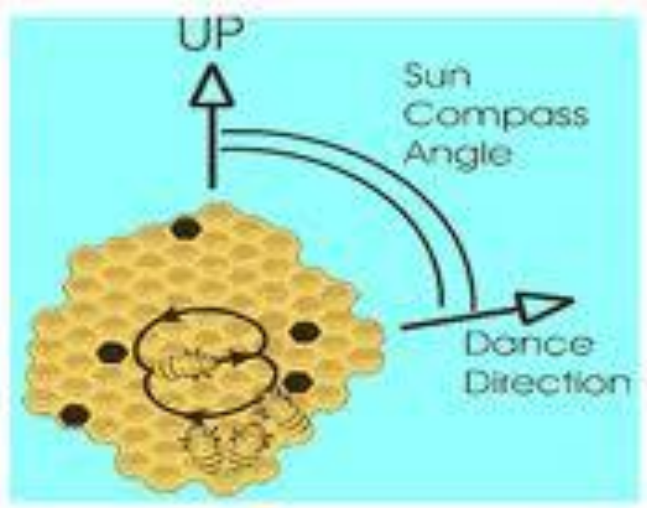
Sense of Smell – Ant Colony Optimization: Bio-inspired algorithm for searching Relationships in Social Networks

SWARM TECHNOLOGIES

Bees Algorithm:

Inspiration: It is inspired by the foraging behavior of the honey bees.

Strategy: The objective of the algorithm is to locate and explore good sites within a problem search space.



SWARM TECHNOLOGIES

Bacterial Foraging Optimization Algorithm

- **Inspiration:** It is inspired by the foraging behavior of E.coli bacteria
- **Strategy:** The objective of the algorithm is to allow cells to stochastically and collectively swarm toward optima through a series of three processes:

1. **Chemotaxis**
2. **Reproduction**
3. **Elimination-dispersal**

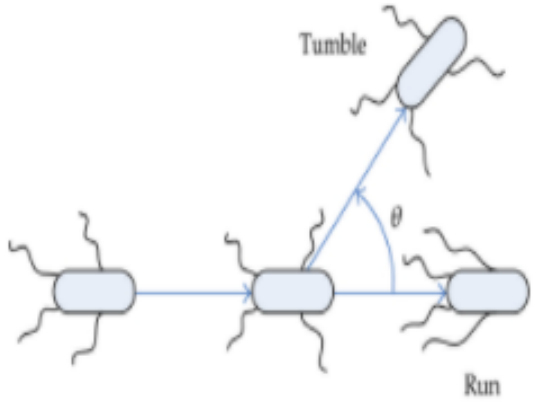


Fig: Chemotactic behavior of E. coli - run and tumble

SWARM TOOLS FOR OPTIMIZATION AND FEATURE EXTRACTION

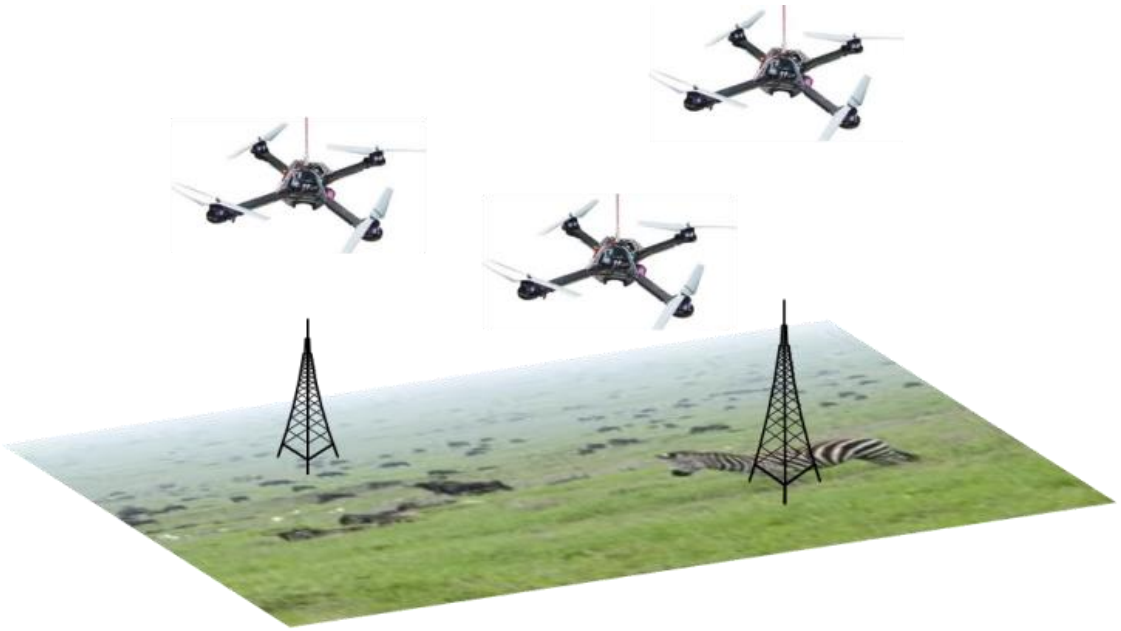


Fig: Route Optimization of Unmanned Aerial Vehicles

SWARM TOOLS FOR OPTIMIZATION AND FEATURE EXTRACTION

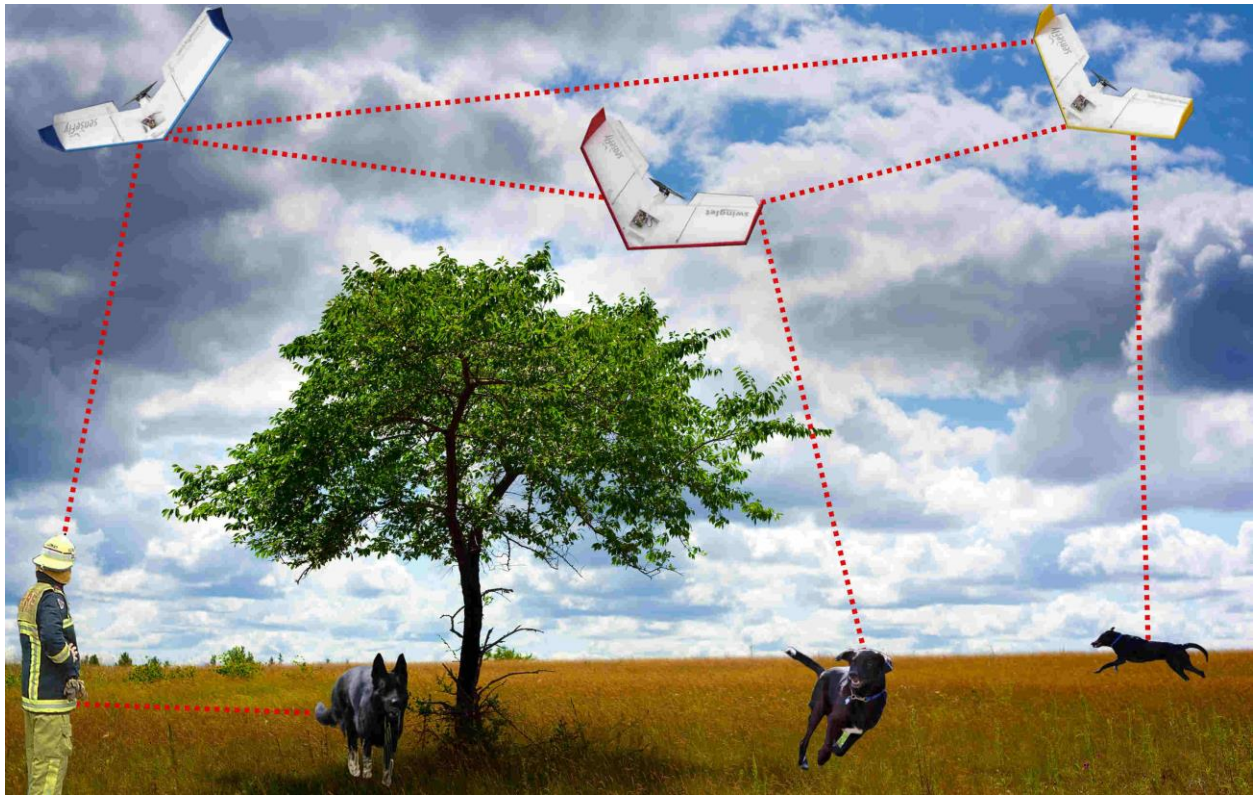


Fig: Distributed Task Allocation in Large, Autonomous, Multirobot Swarm System

SWARM: USE CASES

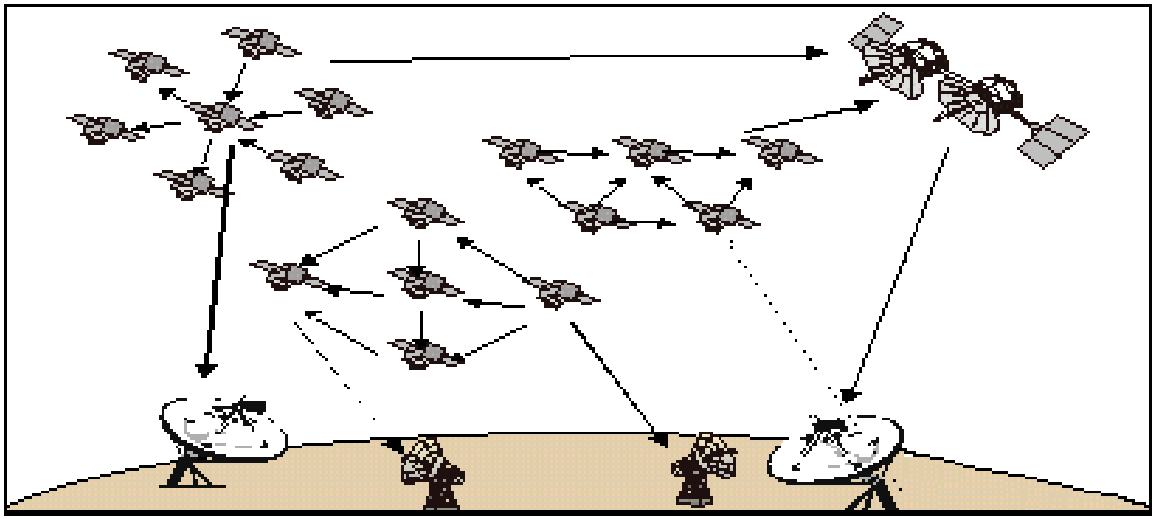


Fig: Micro-satellite Swarm

SWARM: USE CASES

Fig: Swarm Robotics

SWARM: USE CASES

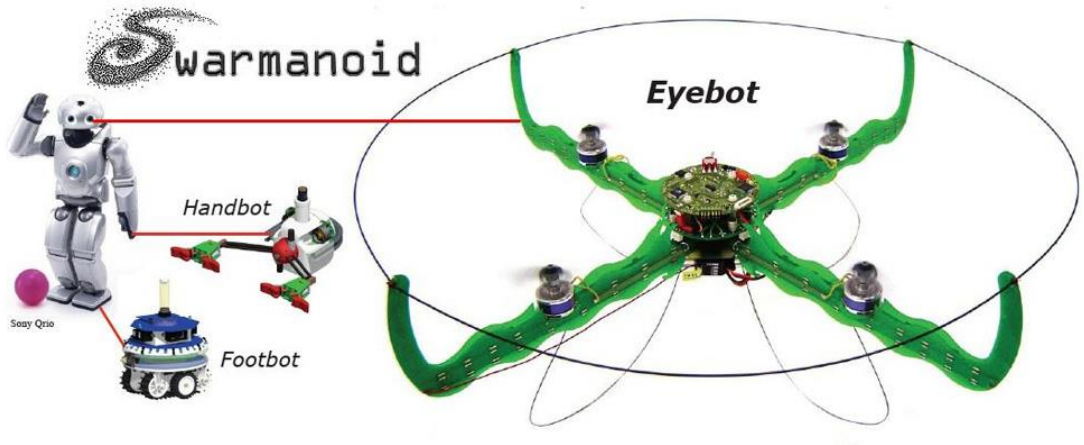
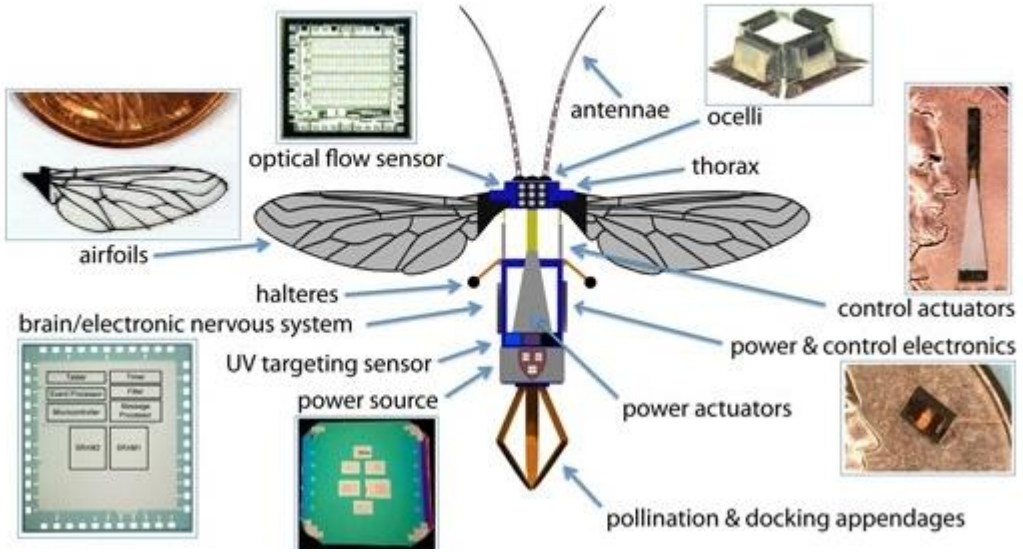


Fig: Swarmanoid: Towards Humanoid Robotic Swarms

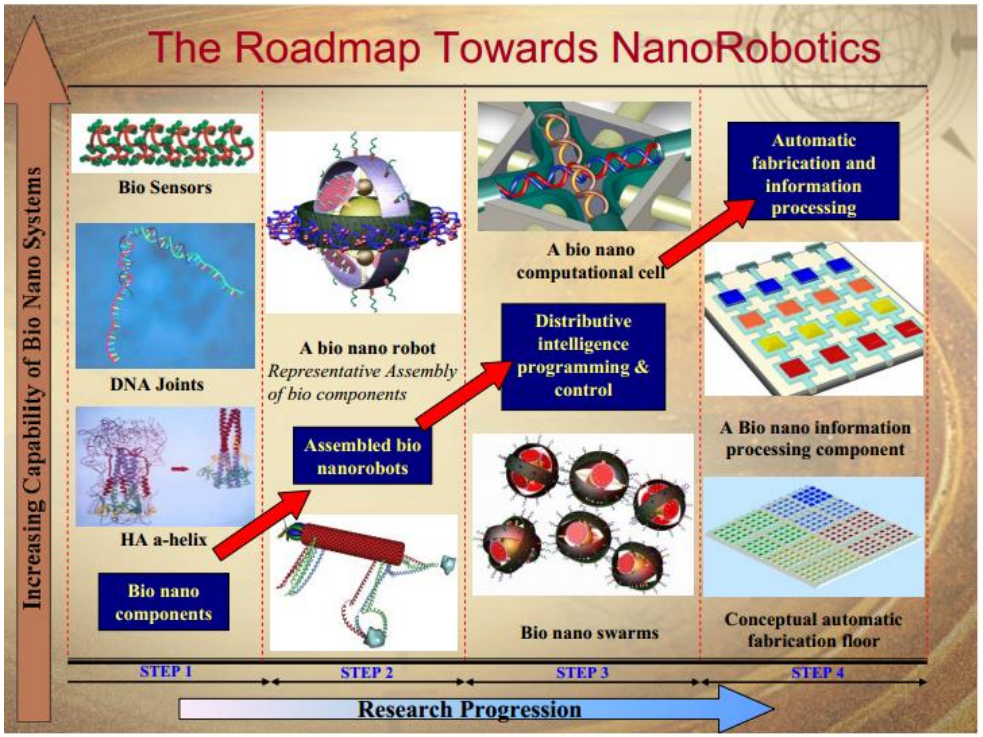
SWARM: USE CASES

Body, Brain, and Colony



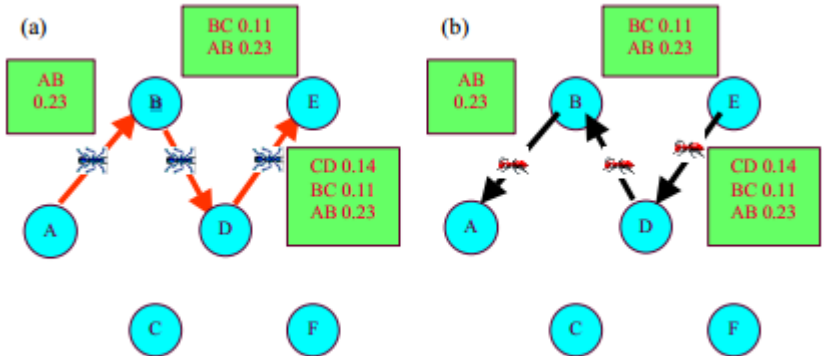
Flight of the Robobee: The Rise of Swarm Robotics

SWARM: USE CASES



Nanorobots / Nanobots / DNA nanotechnology in Medical Applications

SWARM: USE CASES

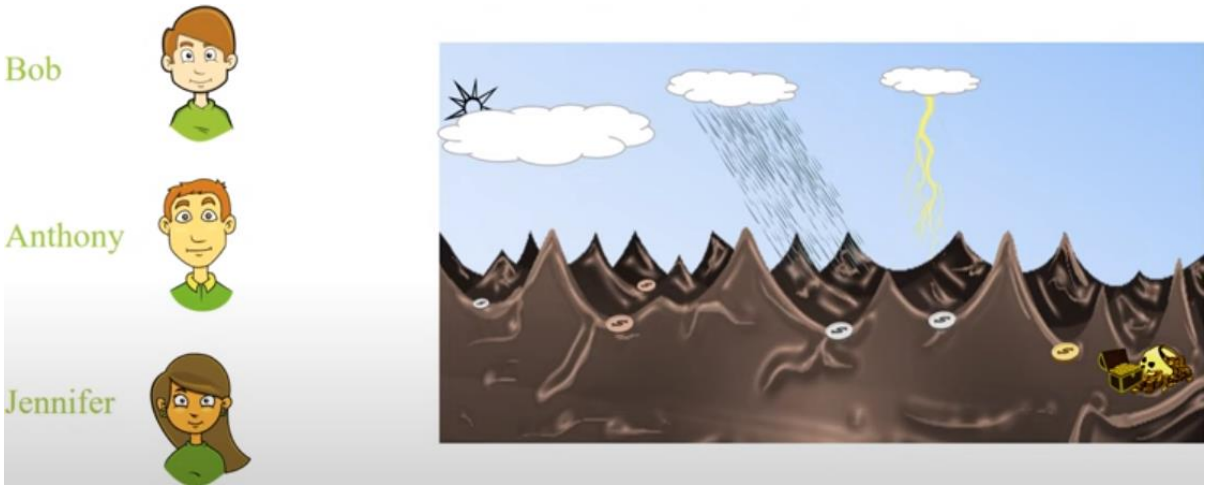


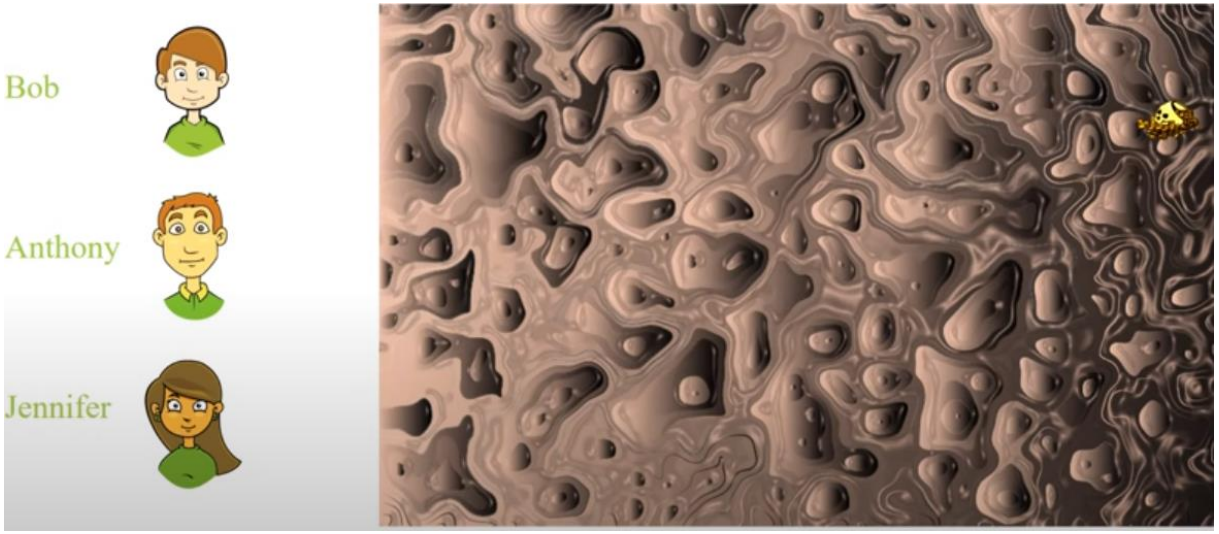
AntNet: Ant-based Swarm Intelligence Algorithm for Routing in Communication Networks

PARTICLE SWARM OPTIMIZATION

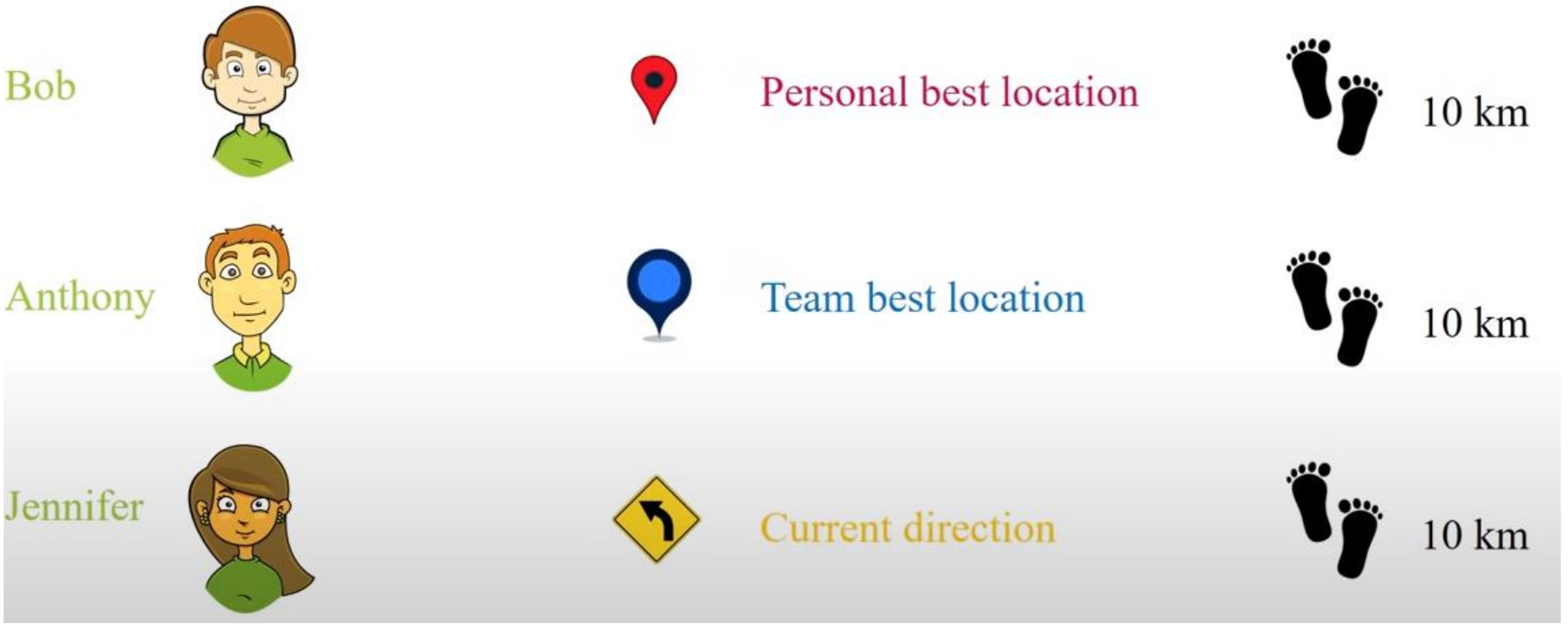
SWARM TECHNOLOGIES

PSO SEARCH STRATEGY

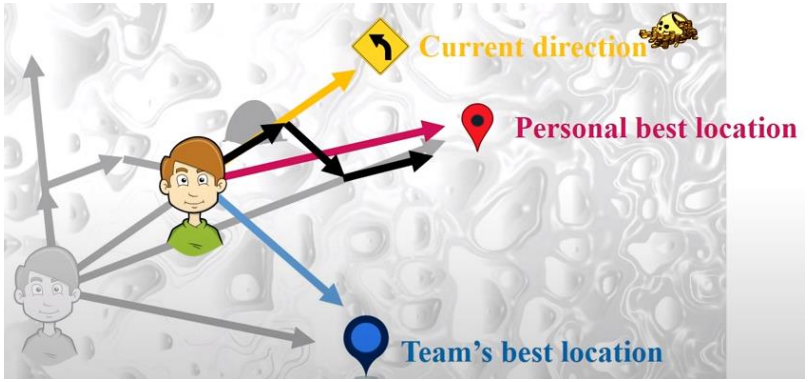




PSO SEARCH STRATEGY

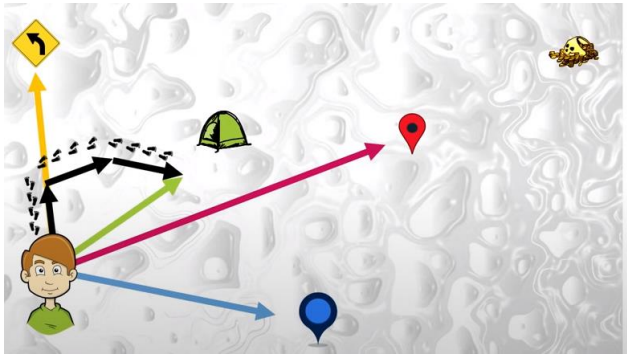


PSO SEARCH STRATEGY

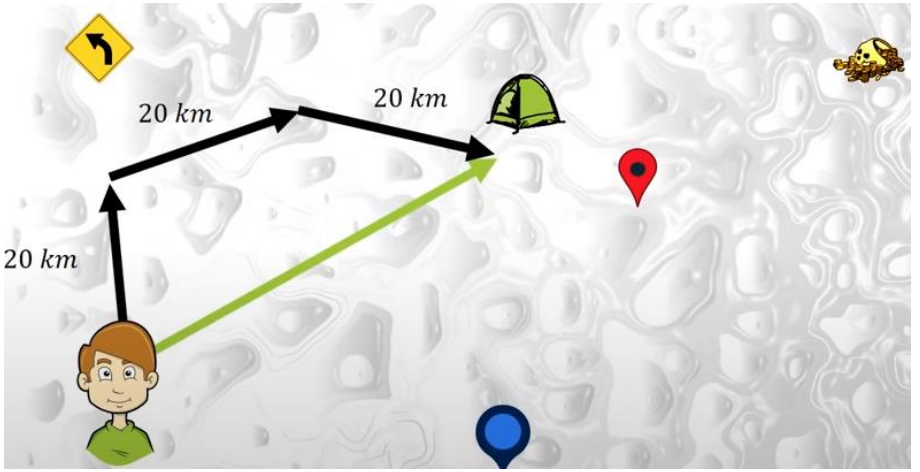


PSO SEARCH STRATEGY

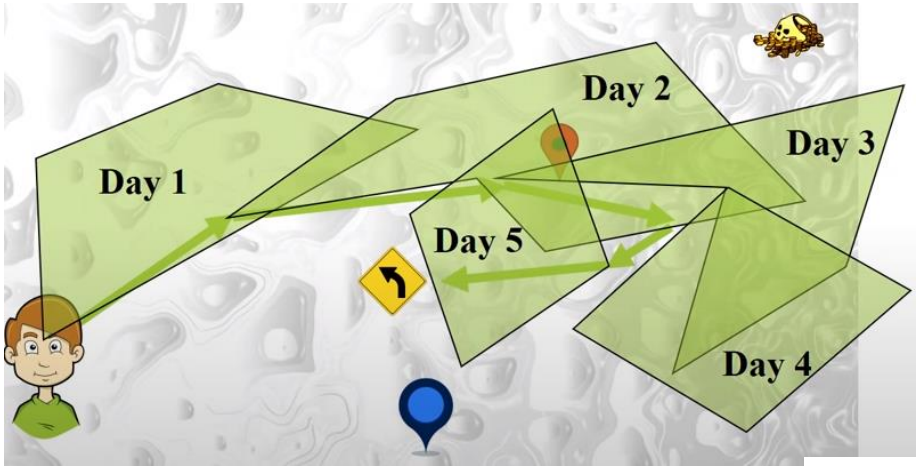
PSO SEARCH STRATEGY

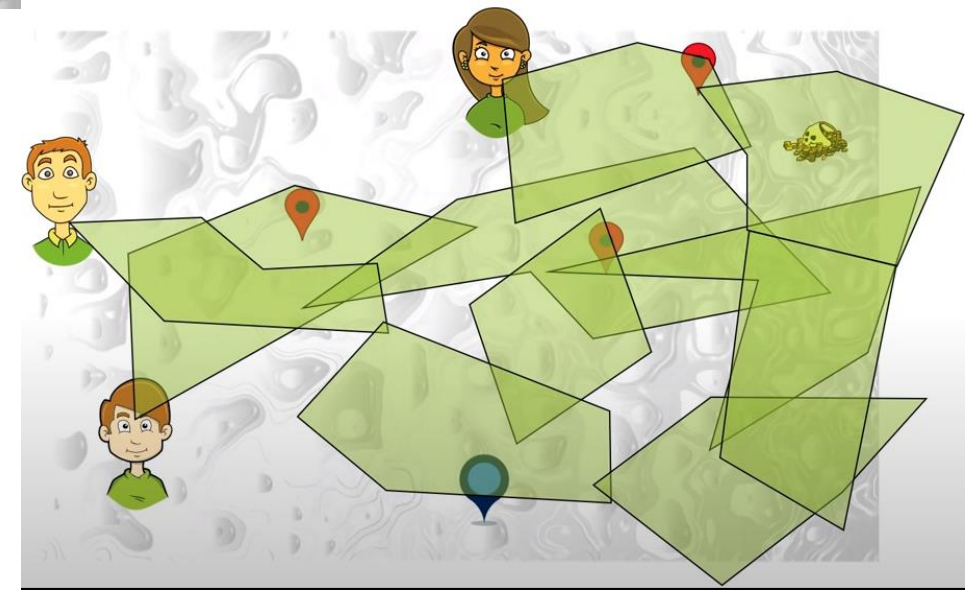


$2 \times r \times 10\ km$
$r\ in\ [0,1]$



PSO SEARCH STRATEGY





PSO SEARCH STRATEGY

Bob 

$$\overrightarrow{X_1^d} = [x_1^d , y_1^d]$$

Anthony 

$$\overrightarrow{X_2^d} = [x_2^d , y_2^d]$$

Jennifer 

$$\overrightarrow{X_3^d} = [x_3^d , y_3^d]$$

$$\overrightarrow{X_i^d} = [x_i^d , y_i^d , z_i^d , \dots]$$

PSO SEARCH STRATEGY

$$\overrightarrow{V_i^{d+1}} = 2r_1\overrightarrow{V_i^d} + 2r_2\left(\overrightarrow{P_i^d} - \overrightarrow{X_i^d}\right) + 2r_3\left(\overrightarrow{G^d} - \overrightarrow{X_i^d}\right)$$

Next velocity (tomorrow)

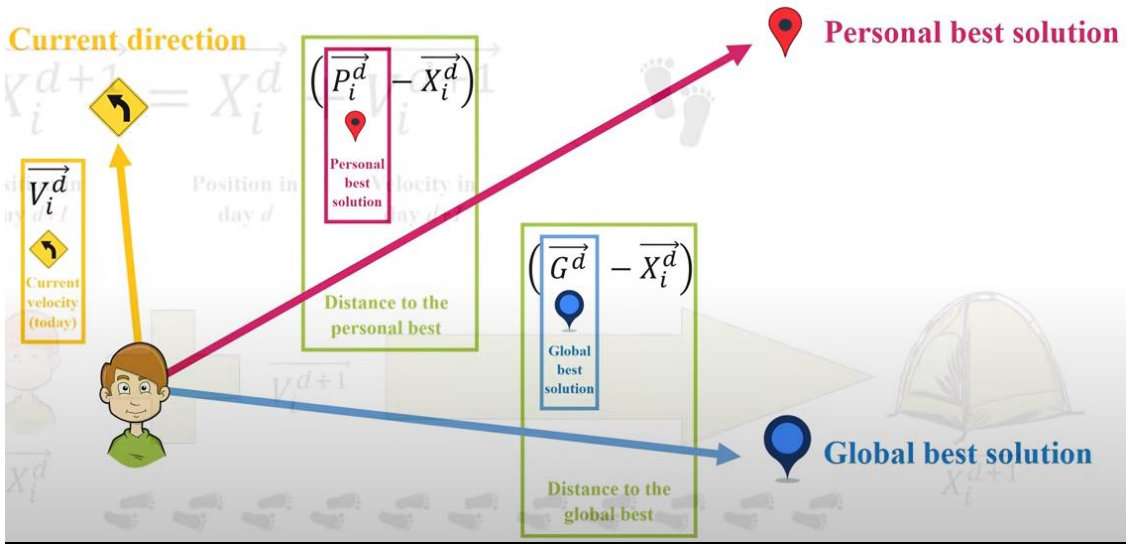
Current velocity (today)

Personal best solution

Distance to the personal best

Global best solution

Distance to the global best



PSO SEARCH STRATEGY

$$\overrightarrow{X_i^{d+1}} = \overrightarrow{X_i^{d}} + \overrightarrow{V_i^{d+1}}$$

Position in day *d+1* | Position in day *d* | Velocity in day *d+1*

$$\overrightarrow{X_i^{t+1}} = \overrightarrow{X_i^{t}} + \overrightarrow{V_i^{t+1}}$$

$$\overrightarrow{V_i^{t+1}} = w\overrightarrow{V_i^{t}} + c_1 r_1 \left(\overrightarrow{P_i^{t}} - \overrightarrow{X_i^{t}}\right) + c_2 r_2 \left(\overrightarrow{G^{t}} - \overrightarrow{X_i^{t}}\right)$$

Inertia | Cognitive component | Social component

PSO ALGORITHM - FUNDAMENTALS

- In PSO, each member of the population is called a **Particle** and population is called a **swarm**.
- How to evaluate the fitness values for each particle?
- What is fitness function?
- What is Optimization?
- What is the significance of Optimization?
- Why objective function is used?

PSO ALGORITHM – NUMERICAL EXAMPLE

Algorithm

STEP 1. Initialization

Initialize Parameters

Initialize Population

- Initialize Position ($\boldsymbol{x_i}$) Randomly for each Particle
- Initialize Velocity ($\boldsymbol{v_i}$) Randomly for each Particle

PSO ALGORITHM – NUMERICAL EXAMPLE

Algorithm

STEP 2. Evaluate Fitness f(x_i^t)

Calculate Fitness Value for Each Particle

If **Fitness Value is better than Best Fitness value (*gBest*)**

Than

Set New value as new (*gBest*)

Choose Particle with Best Fitness Value as *gBest*

PSO ALGORITHM – NUMERICAL EXAMPLE

Algorithm

STEP 3. For Each Particle calculate Velocity and Position.

- Calculate Particles Position by : $x_i^{t+1} = x_i^t + v_i^t * t$
- Calculate Velocity by: $v_{k+1}^i = wv_k^i + c_1 r_1 (xBest_i^t - x_i^t) + c_2 r_2 (gBest_i^t - x_i^t)$

STEP 4. Evaluate Fitness f(x_i^t)

Find Current Best [*gBest*]

STEP 5. Update t=t+1

STEP 6. Output *gBest* & x_i^t

PSO ALGORITHM – NUMERICAL EXAMPLE

Objective function Used for Minimization:

Fun = 10 * (x1 – 1) ^ 2 + 20 * (x2 – 2)^2 + 30 * (x3 – 3)^2

STEP 1. Initialization

Initialize Parameters

Number of Variables: m = 3;

Population Size: n = 5;

Inertia weight: Wmax= 0.9, Wmin= 0.4;

Acceleration Factor: c1 = 2; c2 = 2;

Maximum Iteration Size: Maxt= 50;

COND…

STEP 1. Initialization

- Initialize Velocity (v_i) Randomly for each Particle
- v = 0.1 * x0(i, j);

Initial Velocity for First Particle

v(x1) = 0.1 * 8 =0.8

v(x2) = 0.1 * 9 = 0.9

v(x3) = 0.1 * 1 = 0.1

	x1	x2	x3	
1st Particle	8	9	1	
2nd Particle	9	6	1	
3rd Particle	3	5	10	
4th Particle	10	2	10	
5th Particle	10	5	8	

Initial Population

	x1	x2	x3	
1st Particle	0.8	0.9	0.1	
2nd Particle	0.9	0.6	0.1	
3rd Particle	0.3	0.5	1	
4th Particle	1	0.2	1	
5th Particle	1	0.5	0.8	

COND…

STEP 1. Initialization

- Initialize Position (x_i) Randomly for each Particle

Current Position = Previous Position + Velocity

Current Position for 1st Particle:

x1 = 8 + 0.8 = 8.8

x2 = 9 + 0.9 = 9.9

x3 = 1 + 0.1 = 1.1

	x1	x2	x3
1st Particle	8.8	9.9	1.1
2nd Particle	9.9	6.6	1.1
3rd Particle	3.3	5.5	11
4th Particle	11	2.2	11
5th Particle	11	5.5	8.8

COND…

STEP 2. Evaluate Fitness $f(x_i^t)$: Calculate Fitness Value for Each Particle

Objective function Used for Minimization:

$$F(x) = 10 * (x1 - 1)\ ^\wedge\ 2 + 20 * (x2 - 2)^\wedge 2 + 30 * (x3 - 3)^\wedge 2$$

$$F(x_1^0) = 10 * (8.8\text{-}1)^\wedge 2 + 20*(9.9\text{-}2)^\wedge 2 + 30*(1.1\text{-}3)^\wedge 2$$

$$F(x_1^0) = 1.9649$$

```
>> x = 10 * (8.8-1)^2 + 20*(9.9-2)^2 + 30*(1.1-3)^2

x =

   1.9649e+03
```

	x1	x2	x3
1st Particle	8.8	9.9	1.1
2nd Particle	9.9	6.6	1.1
3rd Particle	3.3	5.5	11
4th Particle	11	2.2	11
5th Particle	11	5.5	8.8

COND...

$F(x) = 10 * (x1 - 1)\ ^\wedge\ 2 + 20 * (x2 - 2)^\wedge 2 + 30 * (x3 - 3)^\wedge 2$

$F(x_1^0) = 10 * (8.8\text{-}1)^\wedge 2 + 20*(9.9\text{-}2)^\wedge 2 + 30*(1.1\text{-}3)^\wedge 2 = 1.9649$

$F(x_2^0) = 10 * (9.9\text{-}1)^\wedge 2 + 20*(6.6\text{-}2)^\wedge 2 + 30*(1.1\text{-}3)^\wedge 2 = 1.3236$

$F(x_3^0) = 10 * (3.3\text{-}1)^\wedge 2 + 20*(5.5\text{-}2)^\wedge 2 + 30*(11\text{-}3)^\wedge 2 = 2.2179$

$F(x_4^0) = 10 * (11\text{-}1)^\wedge 2 + 20*(2.2\text{-}2)^\wedge 2 + 30*(11\text{-}3)^\wedge 2 = 2.9208$

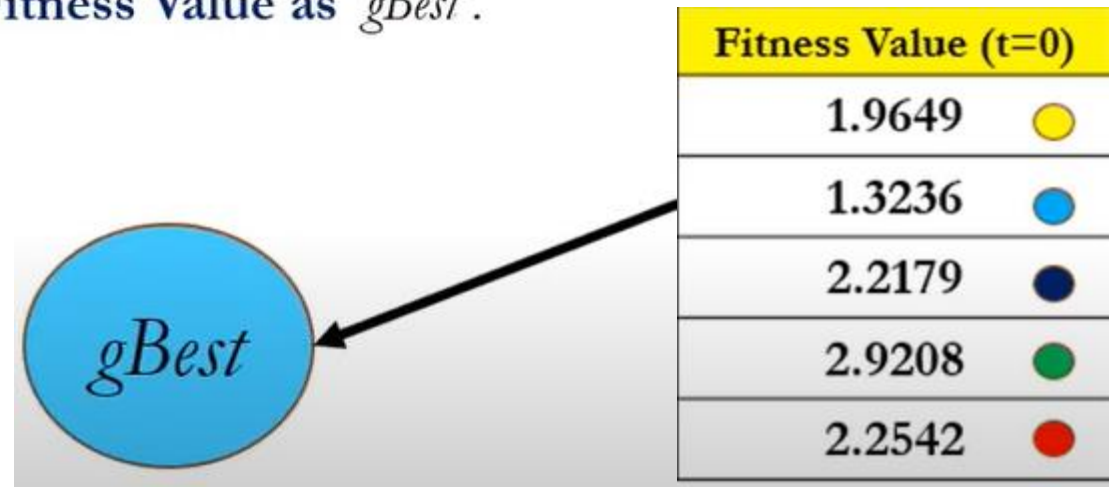
$F(x_5^0) = 10 * (11\ \text{-}1)^\wedge 2 + 20*(5.5\text{-}2)^\wedge 2 + 30*(8.8\text{-}3)^\wedge 2 = 2.2542$

COND…

STEP 2. Evaluate Fitness $f(x_i^t)$

Calculate Fitness Value for Each Particle

Choose Particle with Best Fitness Value as *gBest* .



COND…

STEP 3. For Each Particle calculate Velocity and Position.

- Calculate Particles Position by : $x_i^{t+1} = x_i^t + v_i^t$
- Calculate Velocity by: $v_i^{t+1} = wv_i^t + c_1r_1(xBest_i^t - x_i^t) + c_2r_2(gBest_i^t - x_i^t)$

COND…

STEP 3. For Each Particle calculate Velocity and Position.

- Calculate Velocity by: $v_i^{t+1} = wv_i^t + c_1r_1(xBest_i^t - x_i^t) + c_2r_2(gBest_i^t - x_i^t)$

Update Velocity for First Particle at iteration (t=0)

$$v_1^{0+1} = wv_1^0 + c_1r_1(xBest_1^0 - x_1^0) + c_2r_2(gBest_1^0 - x_1^0)$$

$$v_1^1 = 0.9 * 0.8 + 2 * rand() * (8.8 - 8.8) + 2 * rand() * (9.9 - 8.8) = 1.0667$$

w = 0.9;
c1,c2=2;

	x1	x2	x3
1st Particle	8.8	9.9	1.1
2nd Particle	9.9	6.6	1.1
3rd Particle	3.3	5.5	11
4th Particle	11	2.2	11
5th Particle	11	5.5	8.8

Initial Position

	x1	x2	x3
1st Particle	0.8	0.9	0.1
2nd Particle	0.9	0.6	0.1
3rd Particle	0.3	0.5	1
4th Particle	1	0.2	1
5th Particle	1	0.5	0.8

Initial Velocity

COND…

- Calculate Velocity by: $v_i^{t+1} = wv_i^t + c_1r_1(xBest_i^t - x_i^t) + c_2r_2(gBest_i^t - x_i^t)$

Update Velocity for **First Particle at iteration (t=0)**

$$v_1^{0+1} = wv_1^0 + c_1r_1(xBest_1^0 - x_1^0) + c_2r_2(gBest_1^0 - x_1^0)$$

$v_1^1 = 1.0667$ (x1)

$v_1^1 = -4.4719$ (x2)

$v_1^1 = 0.0900$ (x3)

	x1	x2	x3
1st Particle	8.8	9.9	1.1
2nd Particle	9.9	6.6	1.1
3rd Particle	3.3	5.5	11
4th Particle	11	2.2	11
5th Particle	11	5.5	8.8

Initial Position

	x1	x2	x3
1st Particle	0.8	0.9	0.1
2nd Particle	0.9	0.6	0.1
3rd Particle	0.3	0.5	1
4th Particle	1	0.2	1
5th Particle	1	0.5	0.8

Initial Velocity

COND…

STEP 3. For Each Particle calculate Velocity and Position.

- Calculate Particles Position by : $x_i^{t+1} = x_i^t + v_i^t$

Position updates for 1st Particle:

- $x_1^{0+1} = x_1^0 + v_1^0$
- $x_1^1 = 8.8 + 1.0667 = 9.8667$
- $x_1^1 = 9.9 + (-4.4719) = 5.4281$
- $x_1^1 = 1.1 + 0.900 = 2$

	x1	x2	x3
1st Particle	1.0667	-4.4719	0.0900
2nd Particle	0.8100	0.5400	0.0900
3rd Particle	7.4888	2. 5728	-18.3177
4th Particle	-0.1678	1.4286	1.4286
5th Particle	-1.2109	0.5286	-13.6635

Position updates for each Particle:

	x1	x2	x3
1st Particle	9.8667	5.4281	2
2nd Particle	10.71	7.14	1.19
3rd Particle	10.78	8.07	-7.317
4th Particle	10.83	3.628	12.42
5th Particle	9.78	6.028	-4.8635

Updated Position for each particle

COND…

STEP 4. Objective function Used for Minimization: Calculate Fitness Value

$F(x) = 10 * (x1 - 1)^2 + 20 * (x2 - 2)^2 + 30 * (x3 - 3)^2$

- **First Particle** = 10 * (9.8667-1)^2 + 20*(5.4281-2)^2 + 30*(2-3)^2 = 1.0512

	x1	x2	x3
1st Particle	9.8667	5.4281	2
2nd Particle	10.71	7.14	1.19
3rd Particle	10.78	8.07	-7.317
4th Particle	10.83	3.628	12.42
5th Particle	9.78	6.028	-4.8635

Updated Position for each particle

COND…

STEP 4. Objective function Used for Minimization: Calculate Fitness Value

$F(x) = 10 * (x1 - 1)^2 + 20 * (x2 - 2)^2 + 30 * (x3 - 3)^2$

- **First Particle** = 10 * (9.8667-1)^2 + 20*(5.4281-2)^2 + 30*(2-3)^2 = 1.0512
- 2nd Particle = 10 * (10.71-1)^2 + 20*(7.14-2)^2 + 30*(1.19-3)^2 = 1.5695
- 3rd Particle = 10 * (10.78-1)^2 + 20*(8.07-2)^2 + 30*(-7.317-3)^2 = 4.8866
- 4th Particle = 10 * (10.78-1)^2 + 20*(8.07-2)^2 + 30*(-7.317-3)^2 = 3.6716
- 5th Particle = 10 * (9.78-1)^2 + 20*(6.028-2)^2 + 30*(-4.8635-3)^2 = 2.9504

COND…

Fitness Values Computed as:

Choose the Current Best : Find Current Best [*gBest*]

IF $f(x_1^1) < f(gBest)$ then

$gBest = x_i^t$

$f(x_1^1) < f(gBest)$

$1.0512 < 1.3236$

gBest = 1.0512

$x^0_{1,2,3,4,5}$

Fitness Value (t=0)
1.9649
1.3236
2.2179
2.9208
2.2542

$x^1_{1,2,3,4,5}$

New Fitness Value
1.0512
1.5695
4.8866
3.6716
2.9504

COND…

Choose the Current Best : Find Current Best [*gBest*]

$f(x_2^1) < f(gBest)$

$1.5695 < 1.3236$

$f(x_3^1) < f(gBest)$

$4.8866 < 1.3236$

$f(x_4^1) < f(gBest)$

$3.6716 < 1.3236$

$f(x_5^1) < f(gBest)$

$2.9504 < 1.3236$

Choose the Current Best :Find Particle Best Known Position [*pBest*]

IF f(x_i^t) < f(pBest) then

pBest = x_i^t

f(x_1^1) < f(pBest)

1.0512 < 1.9649

pBest = 1.0512

f(x_2^1) < f(pBest)

1.5695 < 1.3236

f(x_3^1) < f(pBest)

4.8866 > 2.2179

f(x_4^1) < f(pBest)

3.6716 < 2.9208

f(x_5^1) > f(pBest)

2.9504 < 2.2542

$x_{1,2,3,4,5}^0$

Fitness Value (t=0)
1.9649
1.3236
2.2179
2.9208
2.2542

$x_{1,2,3,4,5}^1$

New Fitness Value
1.0512
1.5695
4.8866
3.6716
2.9504

New pBest Value
1.0512
1.3236
7.2179
2.9208
2.2542

pBest [i.e., BEST POSITION] for Each Particle is Updated as

	x1	x2	x3
1st Particle	8.8	9.9	1.1
2nd Particle	9.9	6.6	1.1
3rd Particle	3.3	5.5	11
4th Particle	11	2.2	11
5th Particle	11	5.5	8.8

Initial Position

New pBest Value
1.0512
1.3236
7.2179
2.9208
2.2542

	x1	x2	x3
1st Particle	9.8667	5.4281	2
2nd Particle	9.9	6.6	1.1
3rd Particle	3.3	5.5	11
4th Particle	11	2.2	11
5th Particle	11	5.5	8.8

UPDATED POSITION FOR EACH PARTICLE

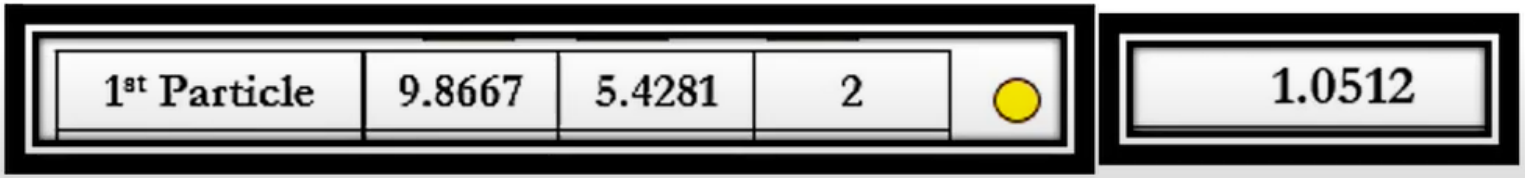
COND…

STEP 5. Update t=t+1

$t = 1 + 1 = 2$

$t = 2$

STEP 6. Output *gBest* & x_i^t



REPEAT UNTIL THE STOPPING CONDITION IS MET

Practice Problem

BACTERIAL FORAGING OPTIMIZATION ALGORITHM

BFOA - INTRODUCTION

- Bacteria Foraging Optimization Algorithm (BFOA), proposed by Passino.
- Application of group foraging strategy of a swarm of E.coli bacteria in multi-optimal function optimization is the key idea of the new algorithm.
- Bacteria search for nutrients in a manner to maximize energy obtained per unit time.
- Individual bacterium also communicates with others by sending signals.
- A bacterium takes foraging decisions after considering two previous factors.
- The process, in which a bacterium moves by taking small steps while searching for nutrients, is called chemotaxis and key idea of BFOA is mimicking chemotactic movement of virtual bacteria in the problem search space.

BASIC DETAILS: BACTERIA FORAGING TECHNOLOGY

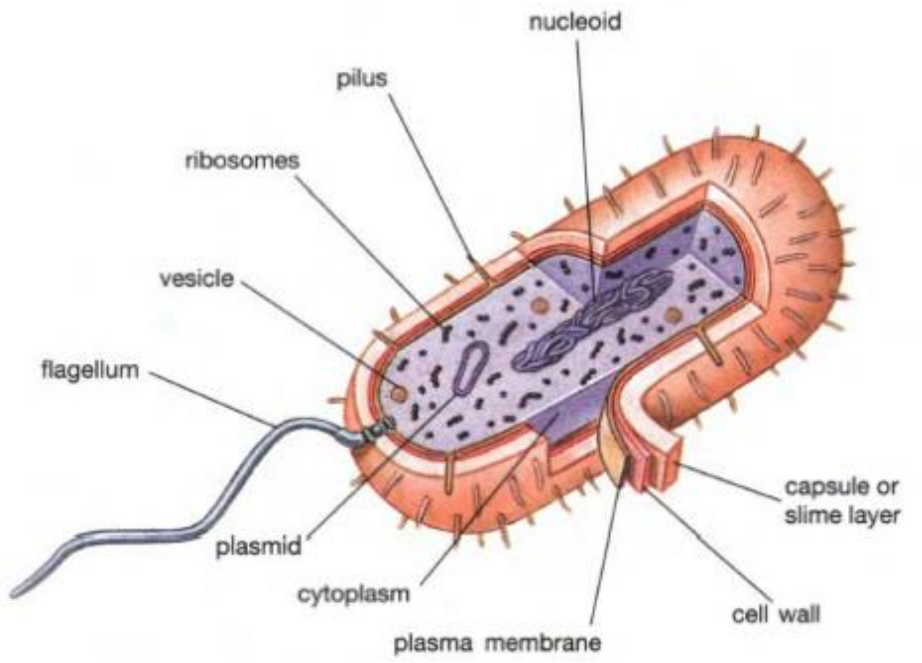
1. Foraging: Element of Foraging Theory

Based on the assumption that animals search for and obtain nutrients in a way that maximizes their energy intake E per unit time T spent foraging.

2. Search Strategies for Foraging

The search strategies of many species are actually between the cruise and ambush extremes.

3. Bacterial Foraging: E.coli



BASIC DETAILS: BACTERIA FORAGING TECHNOLOGY

4. Swimming and Tumbling via Flagella

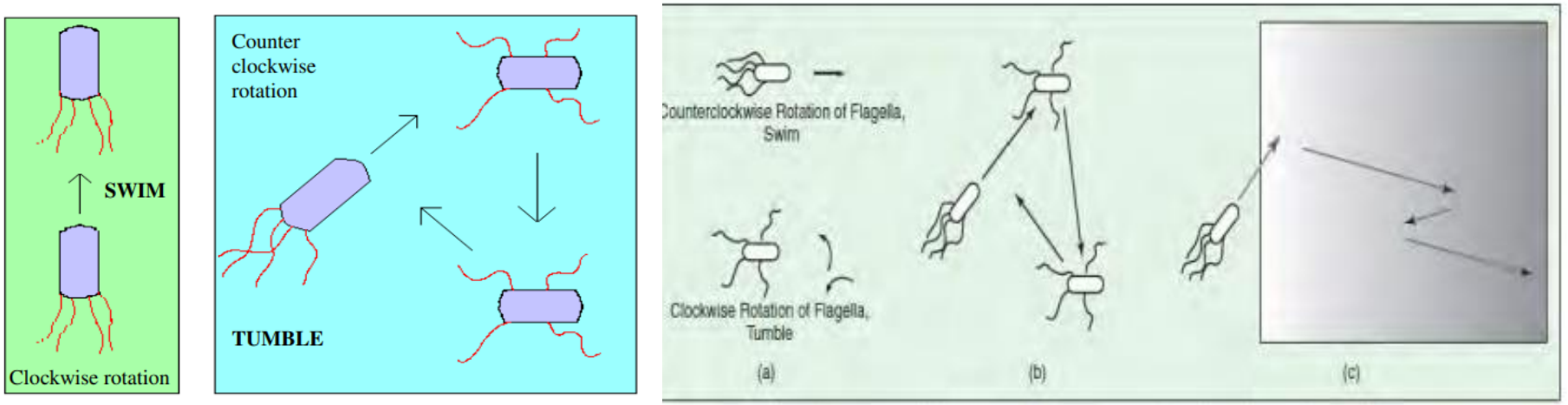


Fig: Swimming, tumbling and chemotactic behavior of E coli

BACTERIAL FORAGING OPTIMIZATION ALGORITHM - STEPS

- **Chemotaxis**

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}, \quad (1)$$

- **Swarming**

$$J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l))$$

$$= \sum_{i=1}^{S}[-d_{\text{attractant}}\exp(-w_{\text{attractant}}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] + \sum_{i=1}^{S}[h_{\text{repellant}}\exp(-w_{\text{repellant}}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] \quad (2)$$

- **Reproduction**

- **Elimination and Dispersal**

ARTIFICIAL BEE COLONY OPTIMIZATION

ARTIFICIAL BEE COLONY OPTIMIZATION

- The **Artificial Bee Colony algorithm** (**ABC**) is an optimization algorithm based on the intelligent foraging behaviour of honey bee swarm, proposed by Derviş Karaboğa (Erciyes University) in 2005.

- The colony consists of three groups of bees:
  - employed bees,
  - onlookers and
  - scouts

In Artificial Bee Colony (ABC) Optimization Algorithm:

1. **Nectar / Food Sources** = Solutions that are associated with each employee bee.
2. **Nectar / Food Source Quality** is determined by their fitness values.
3. Best Solution means solution with greatest **fitness value**.

- Initialization Phase

REPEAT

- Employee Bees Phase
- Onlooker Bees Phase
- Scout Bees Phase
- Memories the best solution achieved.

UNTIL Stopping criteria is met.

ARTIFICIAL BEE COLONY OPTIMIZATION

Step 01: Initialize the population randomly (X_i), i = 1,2,3,4,….Population Size

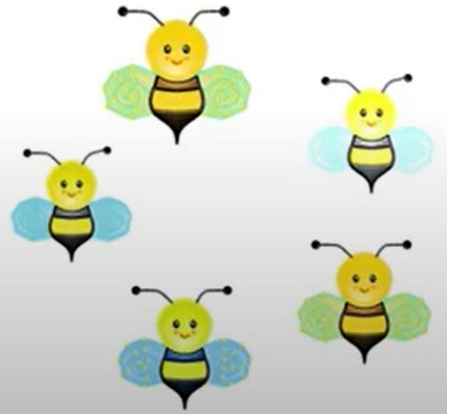
Suppose, Population Size (**Pop**) = 4,

Maximum Number of Iterations (**MaxT**) = 100;

Number of decision Variable (**nVariables**) = 5;

Lower Bound (**LB**) = -10;

Upper Bound (**UB**) = 10;



ARTIFICIAL BEE COLONY OPTIMIZATION

Step 01: Initialize the population randomly (X_i), i = 1,2,3,4,….Population Size

Suppose, Population Size (**Pop**) = 4,

Generate Initial Population for 4 search agents.

Bee No.	Initial Population (X_i)
1.	-1.4512,-0.1857,1.9123,4.5238,-0.2672
2.	-0.9240,-1.3522,6.5063,-8.3306,-7.3366
3.	-6.5322,-2.1812,6.6276,6.0673,-8.7906
4.	-2.0148,0.5375,-1.6640,3.1372,2.5595

ARTIFICIAL BEE COLONY OPTIMIZATION

Step 01: Generate initial population randomly (X_i), i = 1,2,3,4,….Population Size

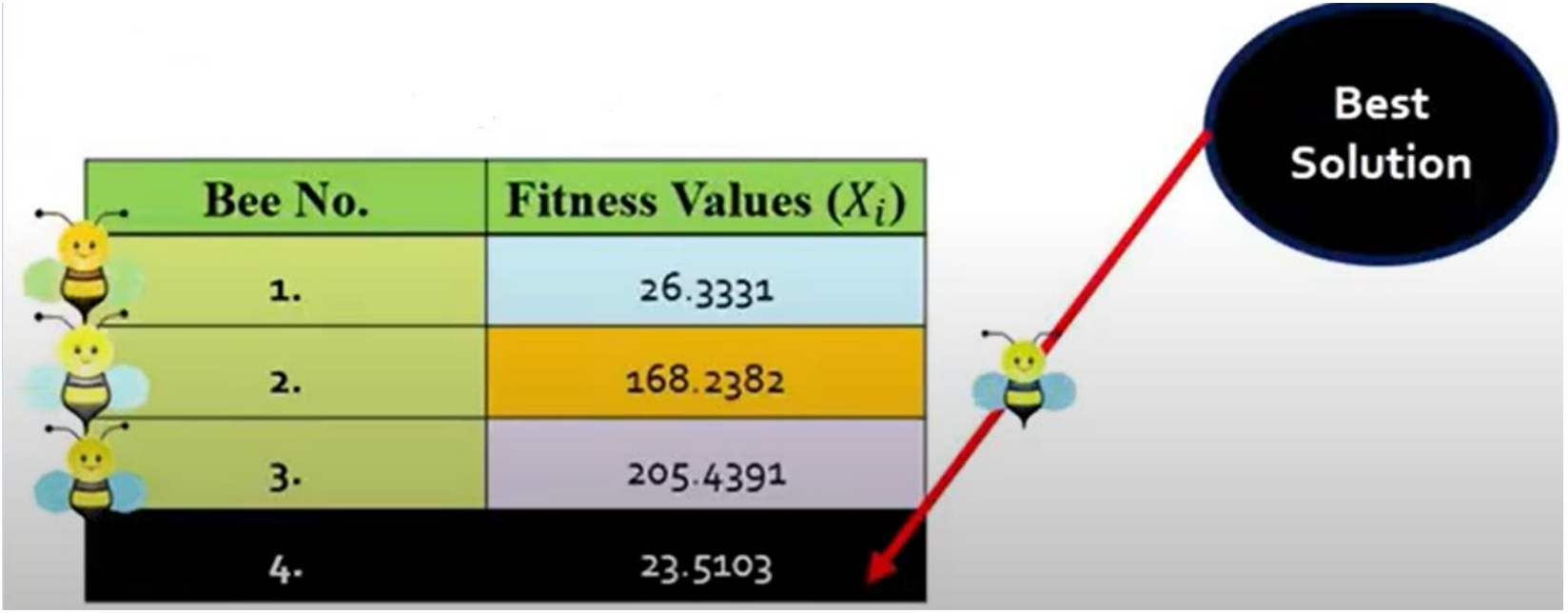
Step 02: Calculate fitness values for each agent in the population

Bee No.	Fitness Values
1	26.3331
2	168.2382
3	205.4391
4	23.5103

Step 01: Generate initial population randomly (X_i), i = 1,2,3,4,….Population Size.

Step 02: Calculate fitness values for each agent in the population.

Step 03: Memorize the best (X_{Best}) solution in the population.



Step 01: Generate initial population randomly (X_i), i = 1,2,3,4,….Population Size.

Step 02: Calculate fitness values for each agent in the population.

Step 03: Memorize the best (X_{Best}) solution in the population.

Step 04: Compute Initial Fitness Vectors.

$$Initial\ Fitness\ Vector_i = \begin{cases} \dfrac{1}{(1+F_i)} & F_i \geq 0 \\ 1 + abs(F_i) & F_i < 0 \end{cases}$$

ARTIFICIAL BEE COLONY OPTIMIZATION

- $Bee\ 1 = \frac{1}{1+Fintness\ Value\ (1)}$
- $Bee\ 1 = \frac{1}{1+26.331}$ $\qquad\qquad \frac{1}{(1+F_i)}$
- Bee 1 = 0.0366

Bee No.	Fitness Values
1	26.3331
2	168.2382
3	205.4391
4	23.5103

ARTIFICIAL BEE COLONY OPTIMIZATION

- $Bee\ 2 = \frac{1}{1+Fintness\ Value\ (2)}$
- $Bee\ 2 = \frac{1}{1+168.2283}$ $\qquad \frac{1}{(1+F_i)}$
- Bee 2= 0.0059

Bee No.	Fitness Values
1	26.3331
2	168.2382
3	205.4391
4	23.5103

ARTIFICIAL BEE COLONY OPTIMIZATION

- $Bee\ 3 = \frac{1}{1+Fintness\ Value\ (3)}$
- $Bee\ 3 = \frac{1}{1+205.4391}$ $\qquad \frac{1}{(1+F_i)}$
- Bee 3= 0.0048

Bee No.	Fitness Values
1	26.3331
2	168.2382
3	205.4391
4	23.5103

ARTIFICIAL BEE COLONY OPTIMIZATION

- $Bee\ 4 = \frac{1}{1+Fintness\ Value\ (4)}$
- $Bee\ 3 = \frac{1}{1+23.5103}$ $\qquad\qquad \frac{1}{(1+F_i)}$
- Bee 4= 0.0408

Bee No.	Fitness Values
1	26.3331
2	168.2382
3	205.4391
4	23.5103

Bee No.	Initial Fitness Vector (X_i)
1.	0.0365
2.	0.0059
3.	0.0048
4.	0.0408

Step 05: Generate new solutions (v_i) for employee bees from old solutions (X_i).

$$v_{i,k} = x_{i,k} + \varphi_{i,k} \times (x_{i,k} - x_{j,k})$$

Bee No.	New Solutions for Employee Bees (v_i)				
1.	-4.2337	-0.6462	1.1297	5.2900	-3.8395
2.	-1.1073	-1.2230	7.1723	-3.1839	-4.3666
3.	-5.5039	-1.8801	7.5063	5.0817	-7.4386
4.	-1.9134	0.1421	-5.3763	-1.3962	0.6415

Step 05: Generate new solutions (v_i) for employee bees from old solutions (X_i).

Step 06: Evaluate Fitness values for new solutions (v_i) .

Bee No.	New Solutions Fitness Values (v_i)
1.	62.3435
2.	83.3680
3.	171.3280
4.	34.9473

Step 07: Calculate Fitness vectors for new solutions (v_i) .

Bee No.	New Solutions Fitness Values (v_i)	
1.	62.3435	≥ 0
2.	83.3680	≥ 0
3.	171.3280	≥ 0
4.	34.9473	≥ 0

For Bee 1

$$Bee\ 1 = \frac{1}{1 + Fitness\ Value\ (1)}$$

$$\frac{1}{(1 + F_i)}$$

$$Bee\ 1 = \frac{1}{1 + 62.3435}$$

$$Bee\ 1 = 0.0158$$

Bee No.	New Solutions Fitness Values (v_i)
1.	62.3435
2.	83.3680
3.	171.3280
4.	34.9473

Bee No.	Fitness Vector for New Solutions
1	0.0158
2	0.0118
3	0.0059
4	0.0286

Step 08: Apply the greedy selection process.

- Apply greedy selection between New Solution (v_i) and Old Solutions (X_i).
- If the nectar amount of new solution is Higher than older than bee memories new and forget old solution.

Old (X_1, X_2, X_3, X_4)

New (v_1, v_2, v_3, v_4)

Bee No.	Initial Fitness Vector (X_i)
1.	0.0365
2.	0.0059
3.	0.0048
4.	0.0408

Bee No.	Fitness Vector for new solutions (v_i)
1.	0.0158
2.	0.0118
3.	0.0059
4.	0.0286

Step 05: Generate new solutions (v_i) from old solutions (X_i).

%For Each Employee Bee (X_i) generate new solutions (v_i):

$$v_{i,k} = x_{i,k} + \varphi_{i,k} \times (x_{i,k} - x_{j,k})$$

If(new candidate solution is generated)

{

Perform Greedy Selection.

}

If (Fitness value for v_i is better than X_i)

{

update X_i with v_i.

}

Else{

Keep X_i unchanged.

}

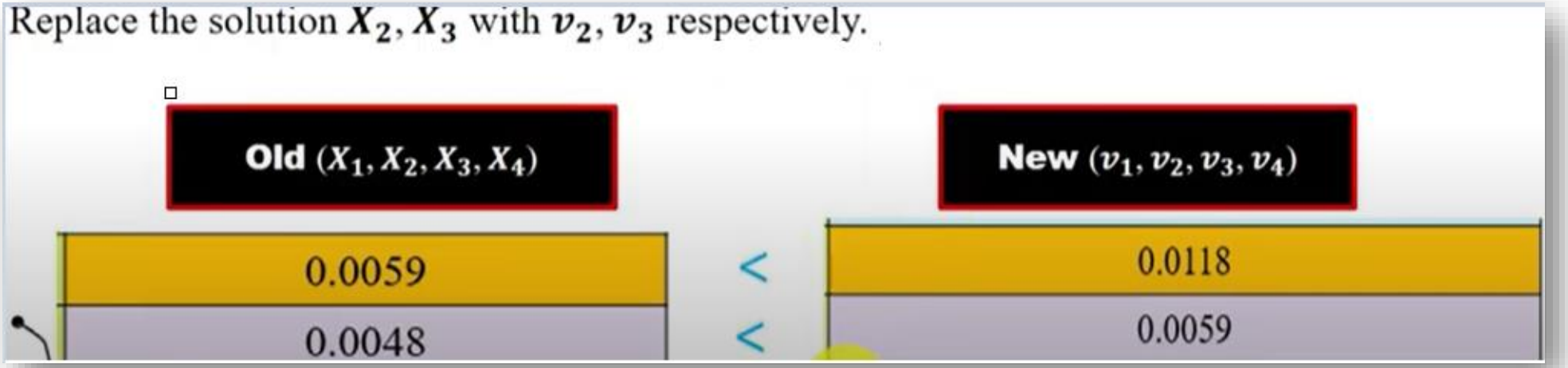
Step 08: Apply the greedy selection process.

- Apply greedy selection between New Solution (v_i) and Old Solutions (X_i).
- If the nectar amount of new solution is Higher than older than bee memories new and forget old solution.

Old (X_1, X_2, X_3, X_4) **New** (v_1, v_2, v_3, v_4)

Initial Fitness Vector (X_i)		Fitness Vector for new solutions (v_i)
0.0365	>	0.0158
0.0059	<	0.0118
0.0048	<	0.0059
0.0408	>	0.0286

Replace the solution X_2, X_3 with v_2, v_3 respectively.



Step 08: Apply the greedy selection process.

- Updated Solutions after greedy selection.

Bee No.	Initial Population (X_i)
1.	-1.4512,-0.1857,1.9123,4.5238,-0.2672
2.	
3.	
4.	-2.0148,0.5375,-1.6640,3.1372,2.5595

Bee No.	Initial Fitness Vector (X_i)
1.	0.0365
2.	
3.	
4.	0.0408

Bee No.	Fitness Values
1.	26.3331
2.	
3.	
4.	23.5103

Step 08: Apply the greedy selection process.

- Updated Solutions after greedy selection.

Bee No.	Updated Population (X_i)	New Solutions for Employee Bees (v_i)
1.	-1.4512,-0.1857,1.9123,4.5238,-0.2672	-4.2337 -0.6462 1.1297 5.2900 -3.8395
2.	-1.1073,-1.2230,7.1723,-3.1839,-4.3666	-1.1073 -1.2230 7.1723 -3.1839 -4.3666
3.	-5.5039,-1.8801,7.5063,5.0817,-7.4386	-5.5039 -1.8801 7.5063 5.0817 -7.4386
4.	-2.0148,0.5375,-1.6640,3.1372,2.5595	-1.9134 0.1421 -5.3763 -1.3962 0.6415

Step 09: Calculate Probability values (P_i) for solution (X_i)

Onlooker bee task:

- Evaluates the nectar information taken from all employee bees and chooses a food source with a probability related to its nectar amount. Roulette wheel mechanisms is used for probabilistic selection.

$$P_i = \frac{F_i}{\sum_{i=1}^{Npop} F_i}$$

Bee No.	Updated Fitness Vector (X_i)
1.	0.0365
2.	0.0118
3.	0.0059
4.	0.0408

Bee No.	Probability Values(P_i)
1.	0.3842
2.	0.1242
3.	0.0621
4.	0.4294

Better solutions have higher probability of being chosen or exploration.

Step 10: Produce new solutions $(v_{i,k})$ for onlooker bees form previous solution $(X_{i,k})$ selected using probability values.

$$v_{i,k} = x_{i,k} + \varphi_{i,k} \times (x_{i,k} - x_{j,k})$$

Bee No.	Updated Population (X_i)
1.	-1.4512,-0.1857,1.9123,4.5238,-0.2672
2.	-1.1073,-1.2230,7.1723,-3.1839,-4.3666
3.	-5.5039,-1.8801,7.5063,5.0817,-7.4386
4.	-2.0148,0.5375,-1.6640,3.1372,2.5595

Bee No.	Fitness Values
1.	122.9802
2.	87.6232
3.	69.5536
4.	40.2149

Step 11: Apply greedy selection.

Bee No.	Fitness Values
1.	122.9802
2.	87.6232
3.	69.5536
4.	40.2149

Bee No.	Updated Fitness Vector (v_i)
1.	0.0080
2.	0.0114
3.	0.0141
4.	0.0242

Step 12: Apply greedy selection.

Bee No.	Old Updated Fitness Vector (X_i)		Bee No.	New Updated Fitness Vector (v_i)
1.	0.0365	>	1.	00080
2.	0.0118	>	2.	0.0114
3.	0.0059		3.	0.0059
4.	0.0408	>	4.	0.0242

Bee No.	Fitness Vector (X_i)
1.	0.0365
2.	0.0118
3.	0.0059
4.	0.0408

-2.0148,0.5375,-1.6640,3.1372,2.5595	40.2149

Step 13: Increment counter

$t = t + 1;$

Repeat until ($t \leq MaxT$) **stopping criteria met.**

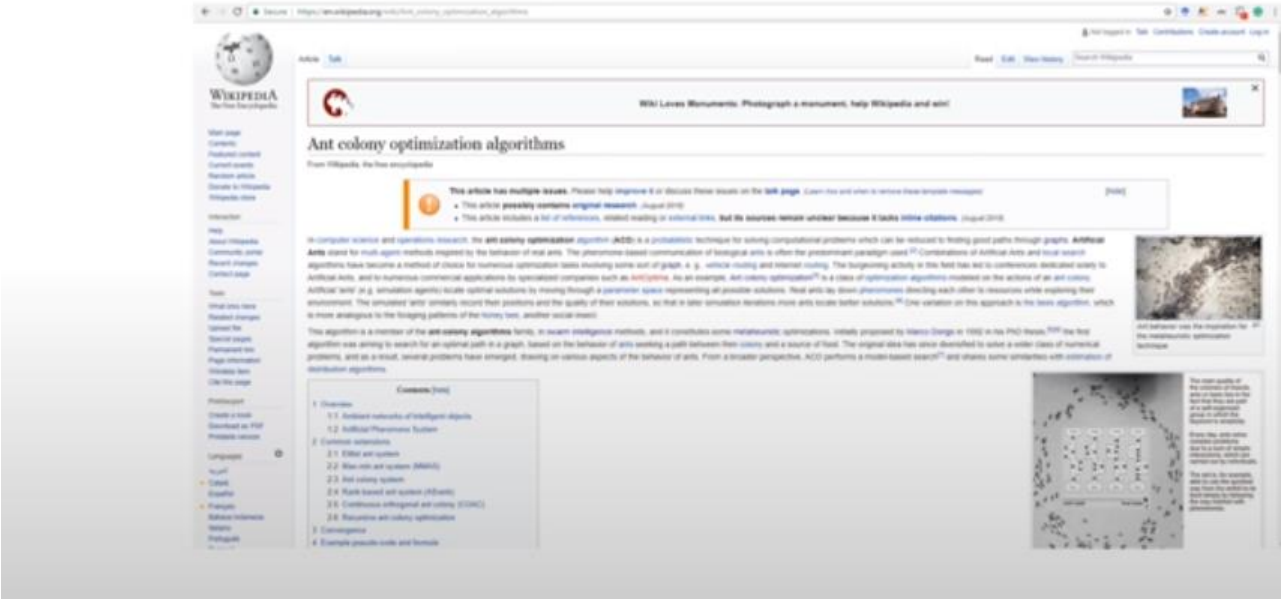
ANT COLONY OPTIMIZATION ALGORITHM

INSPIRATION OF ANT COLONY OPTIMIZATION

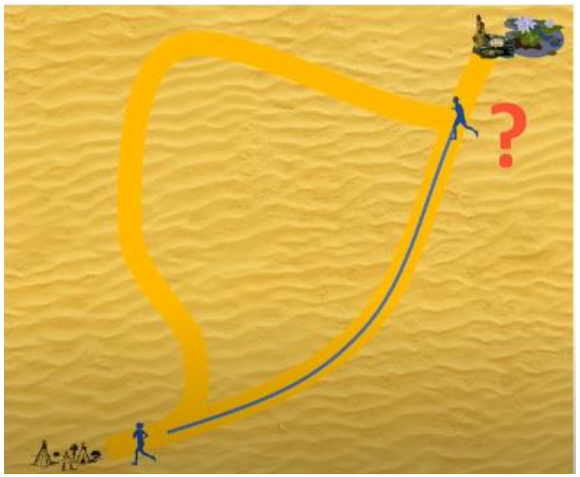
- ACO was proposed by Marco Dorigo in 1992
- The first version of ACO was called Ant Systems
- The main inspiration of the ACO algorithm comes from stigmergy.
- Stigmergy refers to the interaction and coordination of organisms in nature by modifying the environment.

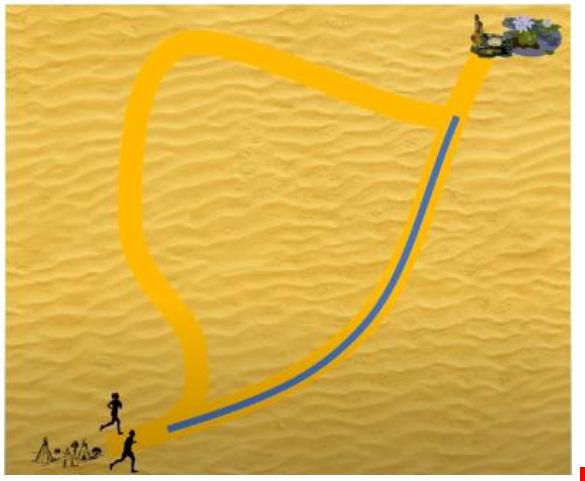
Termite colony

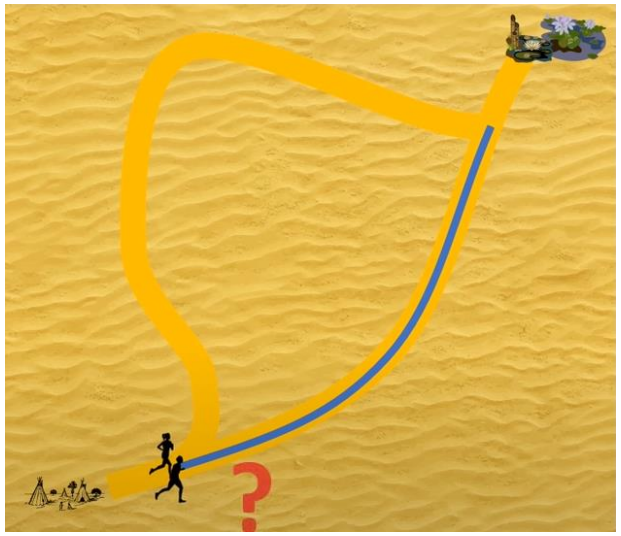
Wikipedia or Reddit websites

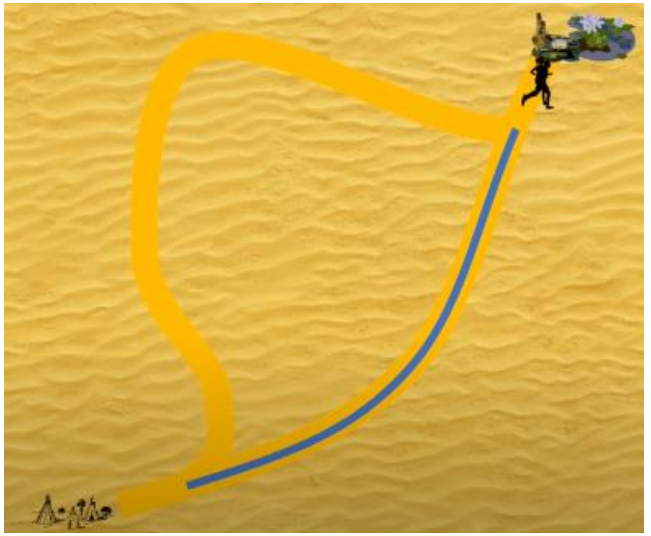


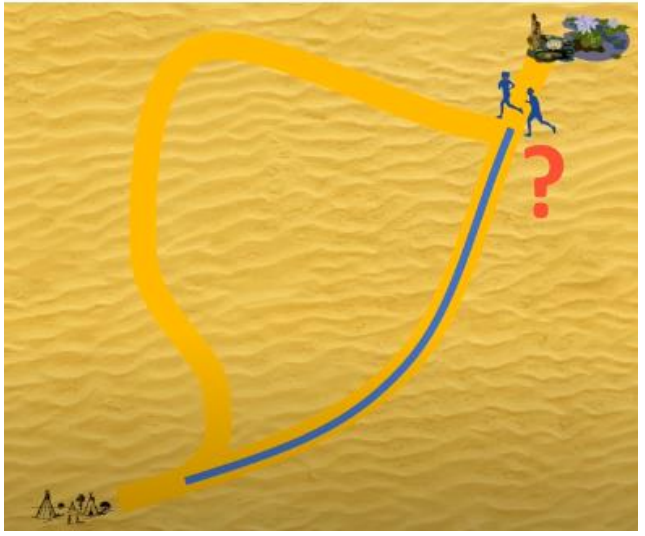
ANALOGY

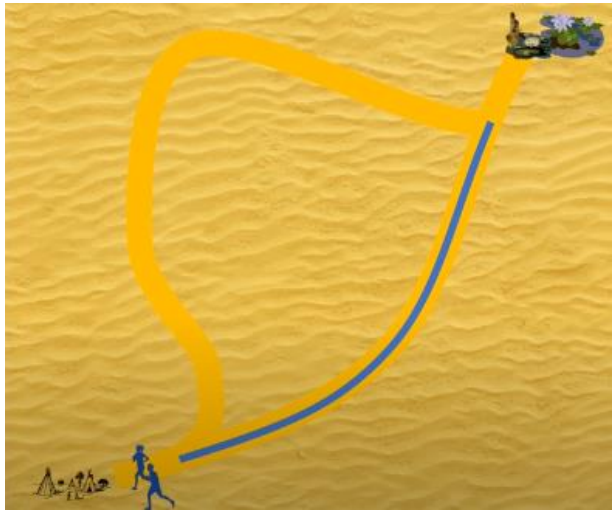




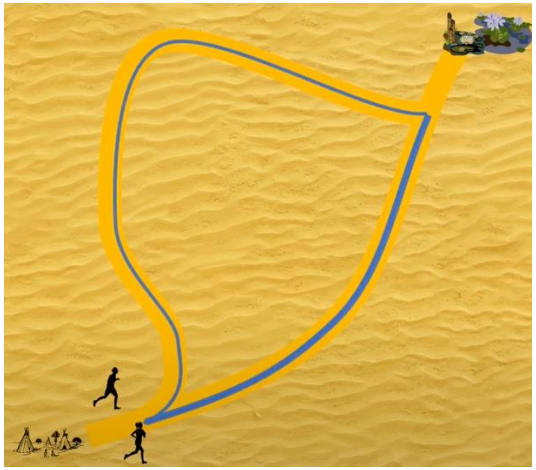


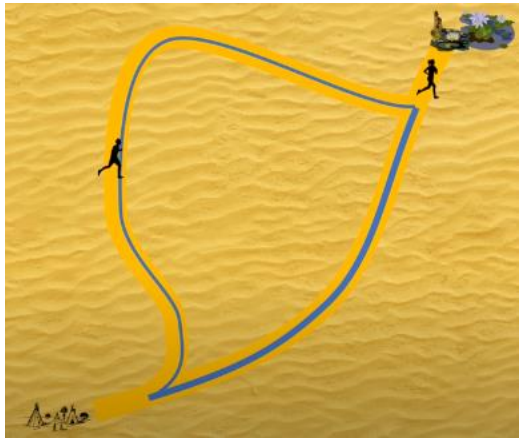


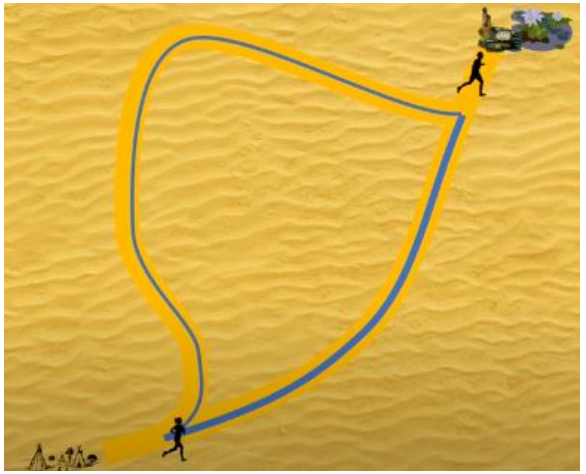


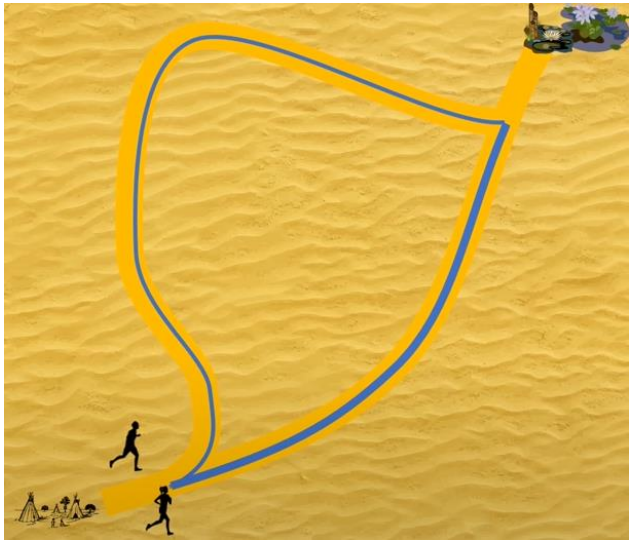


WHAT IF A WRONG PATH IS CHOSEN OR VAPORIZATION OCCURS





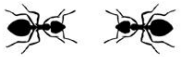




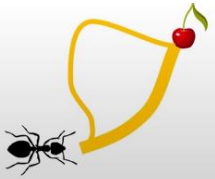
ANTS AND STIGMERGY

- Ants produces chemicals called **pheromone**

- They use **pheromone** to communicate. This is similar to **water in our analogy**.

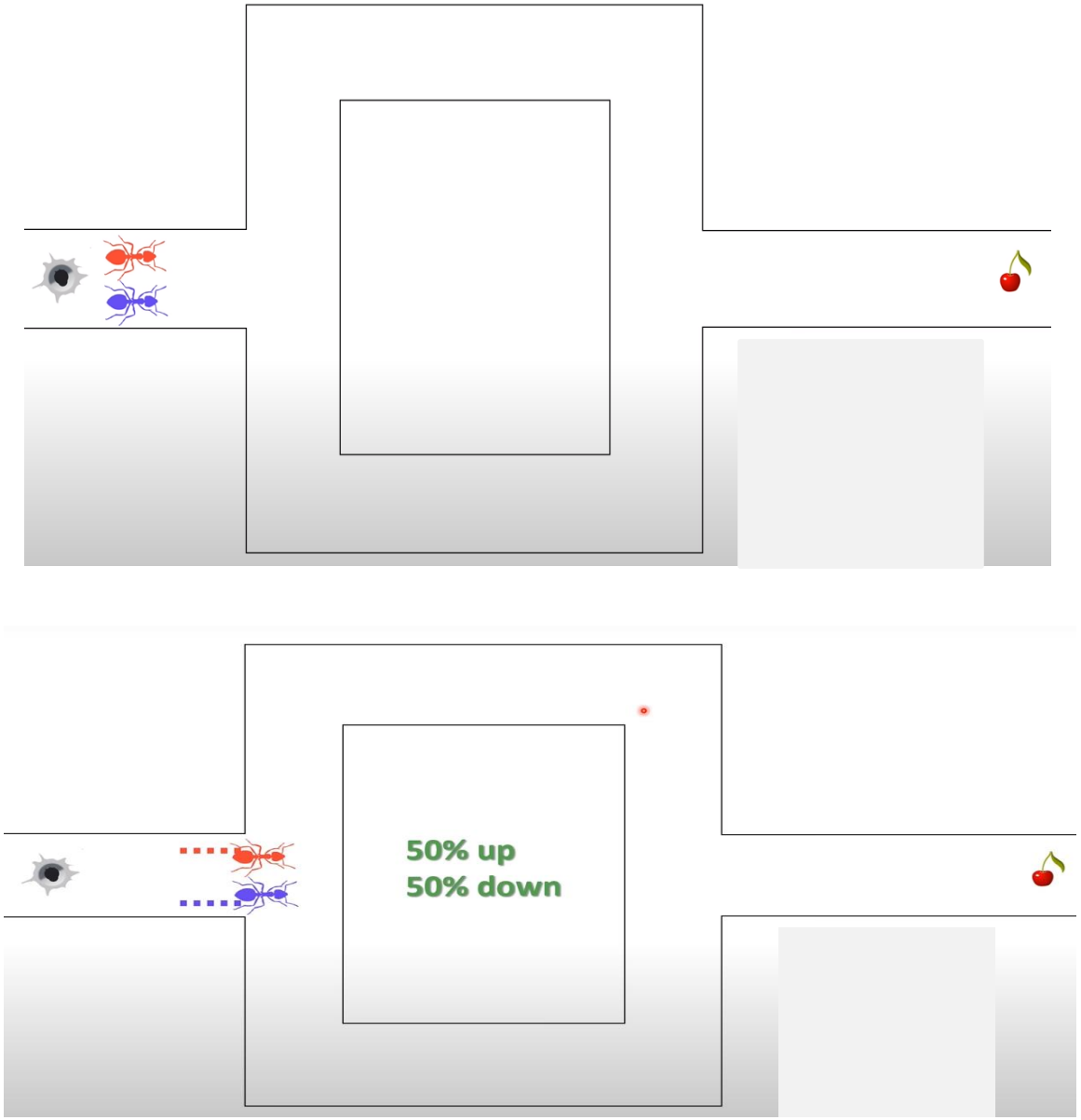


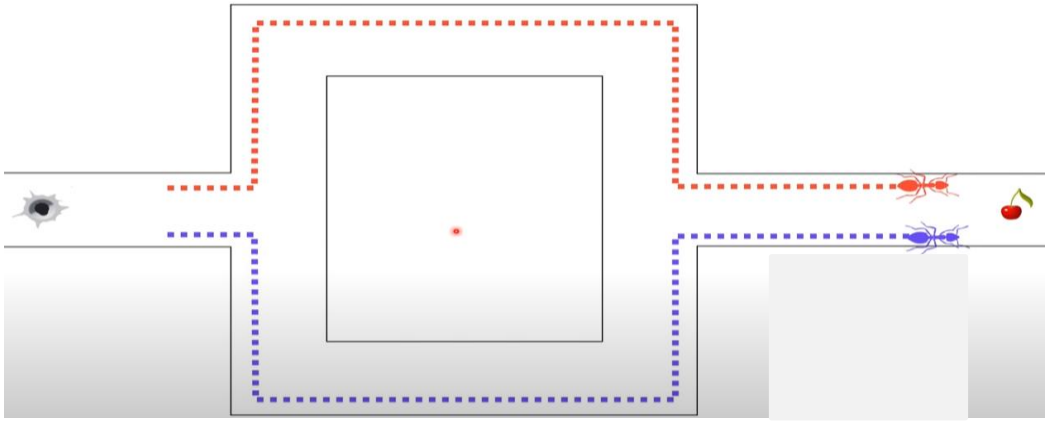
- They use a very similar technique as our analogy to find the shortest path from their nest to a source of food.

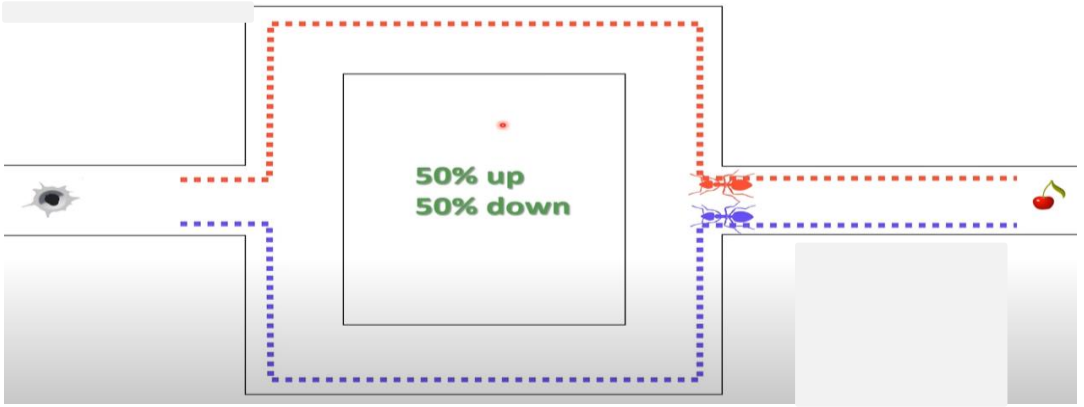


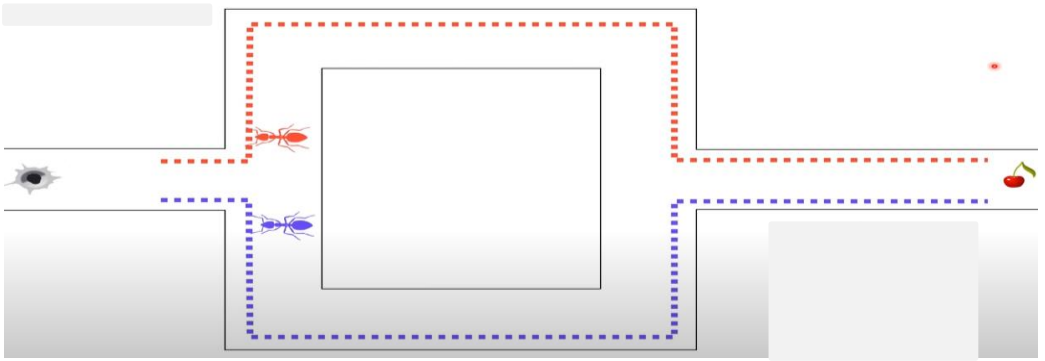
IMPORTANT FACT

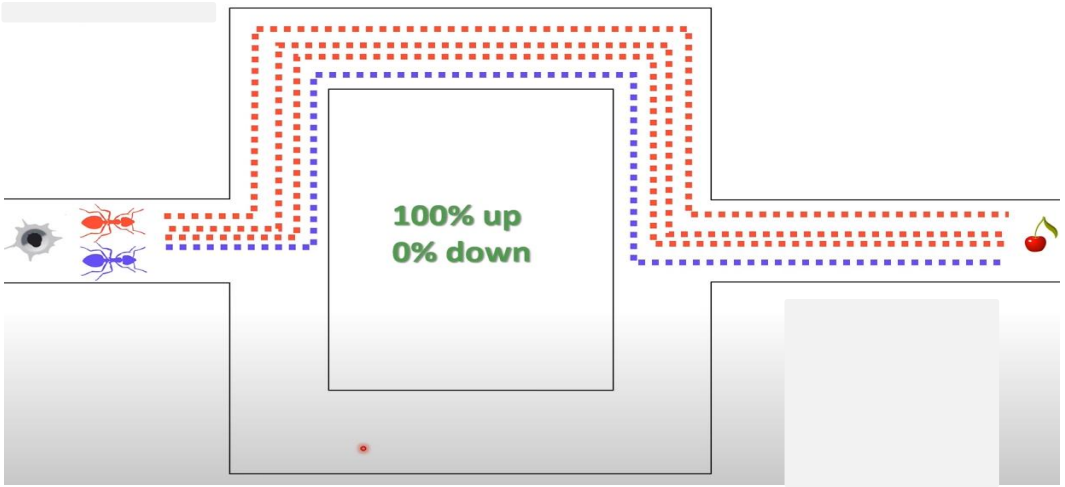
- **Ants are more likely to choose a path with higher pheromone.**
- **This means they make decision based on probabilities.**

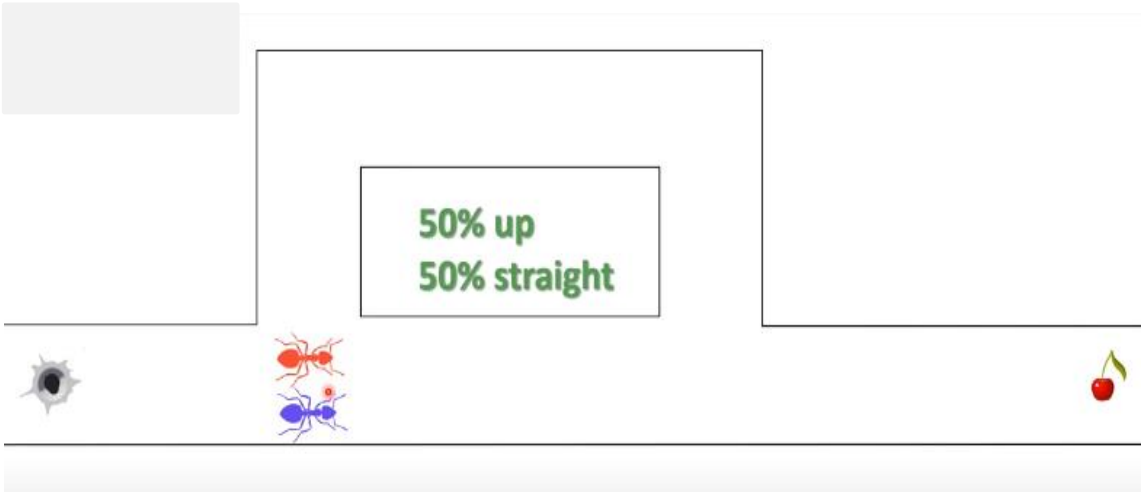


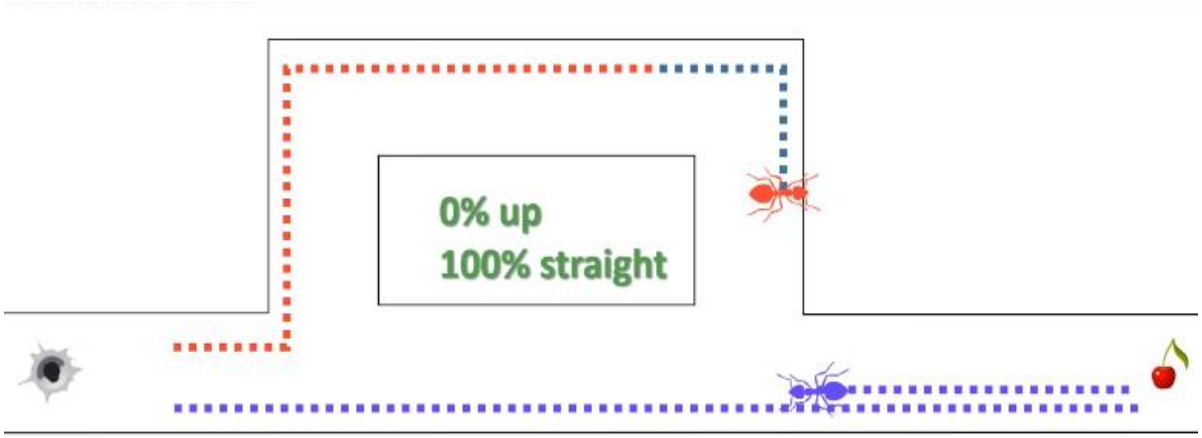


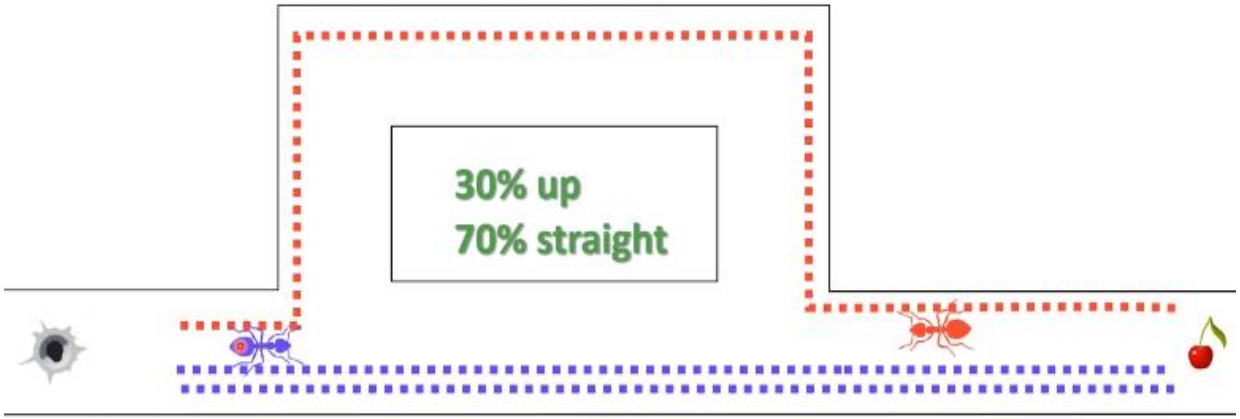


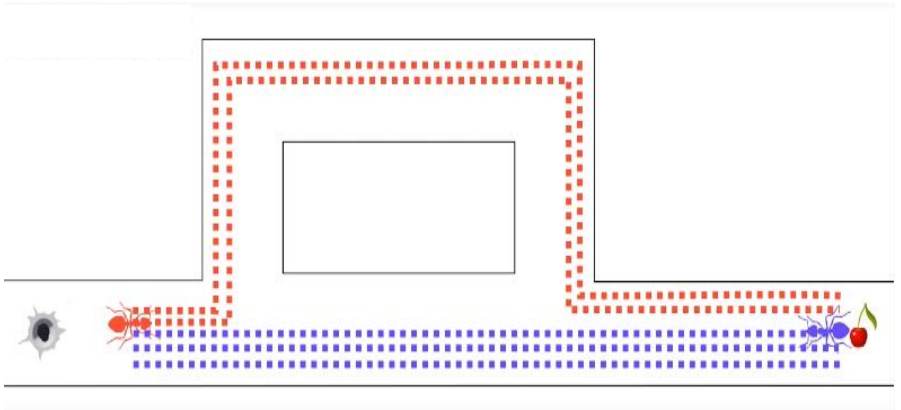


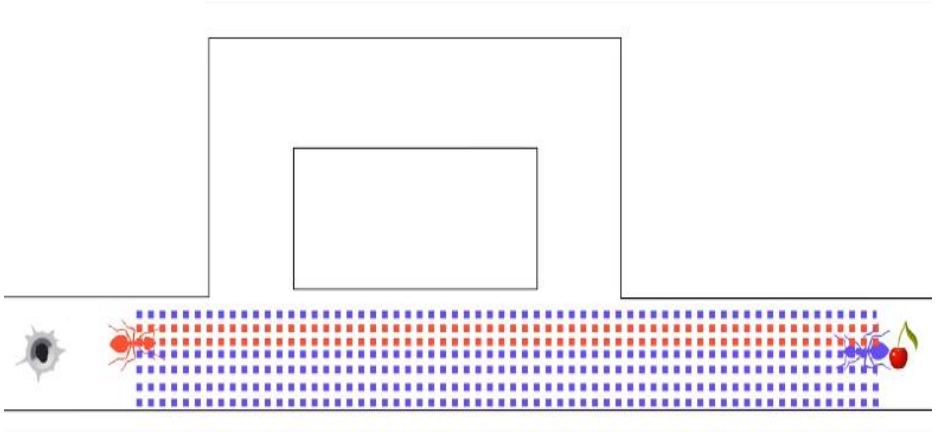










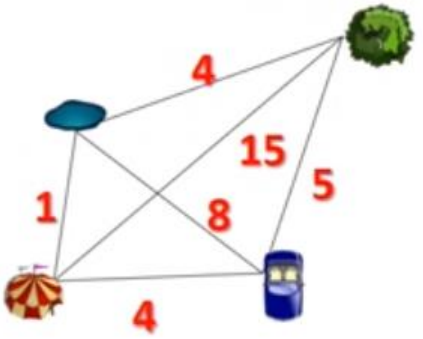


How Ant Colony Algorithm Works

PHEROMONE MATRIX

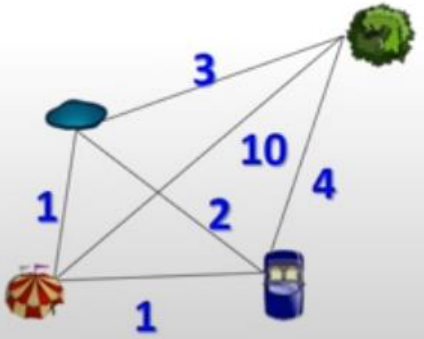
	0	5	15	4
	5	0	4	8
	15	4	0	1
	4	8	1	0

Cost matrix (distance)



	0	4	10	3
	4	0	1	2
	10	1	0	1
	3	2	1	0

Pheromone matrix



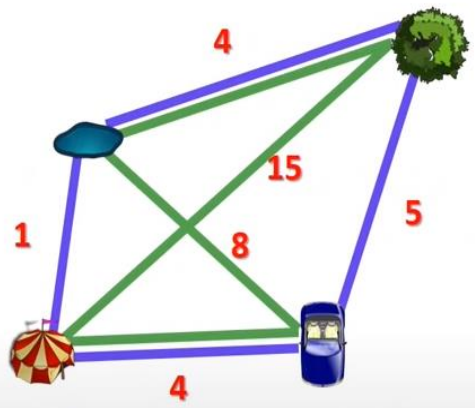
$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{1}{L_k} & k^{th} \text{ ant travels on the edge } i,j \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_{i,j}^{k} = \sum_{k=1}^{m} \Delta\tau_{i,j}^{k} \quad \text{Without vaporization}$$

$$\tau_{i,j}^{k} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k} \quad \text{With vaporization}$$

PHEROMONE MATRIX

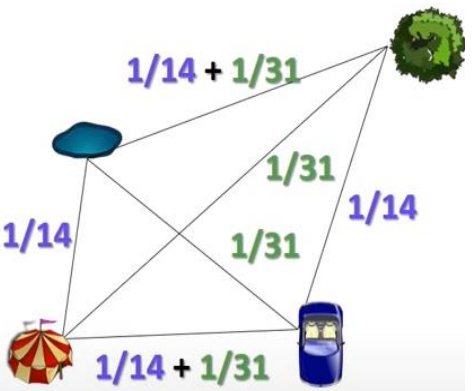
Cost graph



$L_1 = 14 \rightarrow \Delta\tau_{i,j}^1 = \frac{1}{14}$

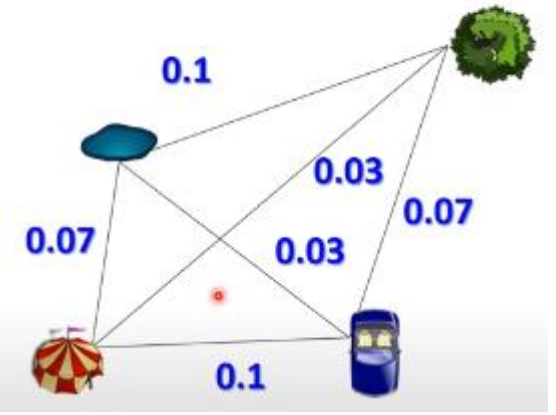
$L_2 = 31 \rightarrow \Delta\tau_{i,j}^2 = \frac{1}{31}$

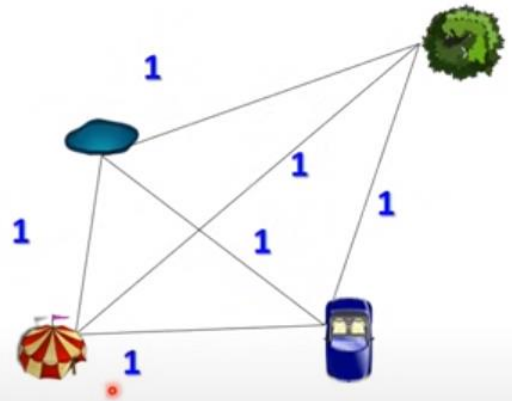
Pheromone graph



$$\tau_{i,j} = \sum_{k=1}^{m} \Delta\tau_{i,j}^k$$

Pheromone graph

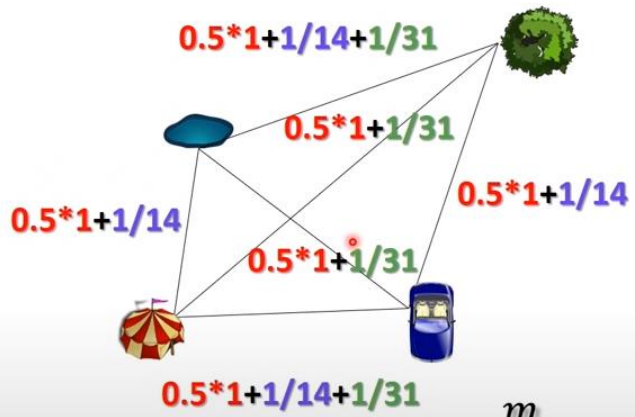




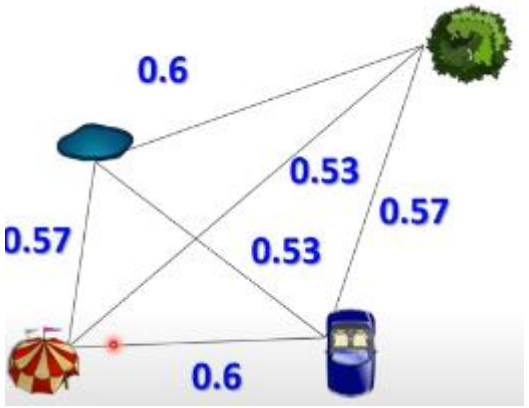
$$\tau_{i,j} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^k$$

$$\rho = 0.5$$

Pheromone graph



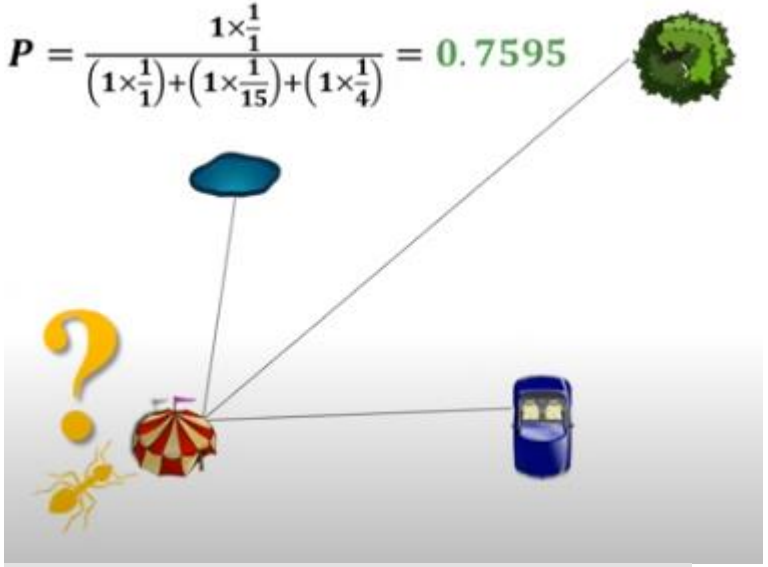
$$\tau_{i,j} = (1-\rho)\,\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}$$

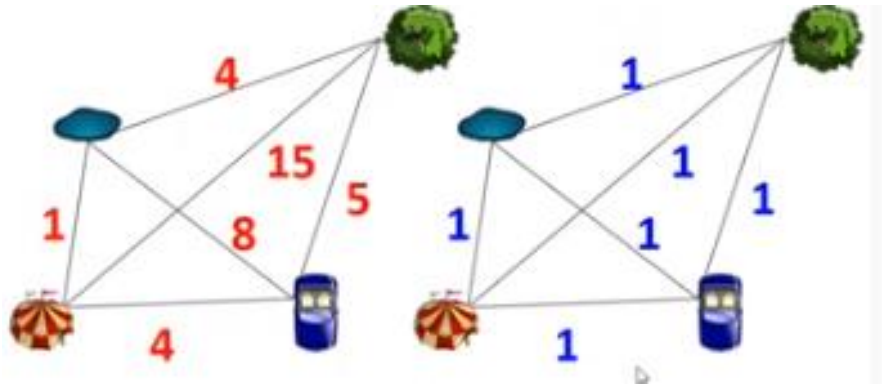


$$P_{i,j} = \frac{(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\sum\left((\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}\right)}$$

$$where: \eta_{i,j} = \frac{1}{L_{i,j}}$$

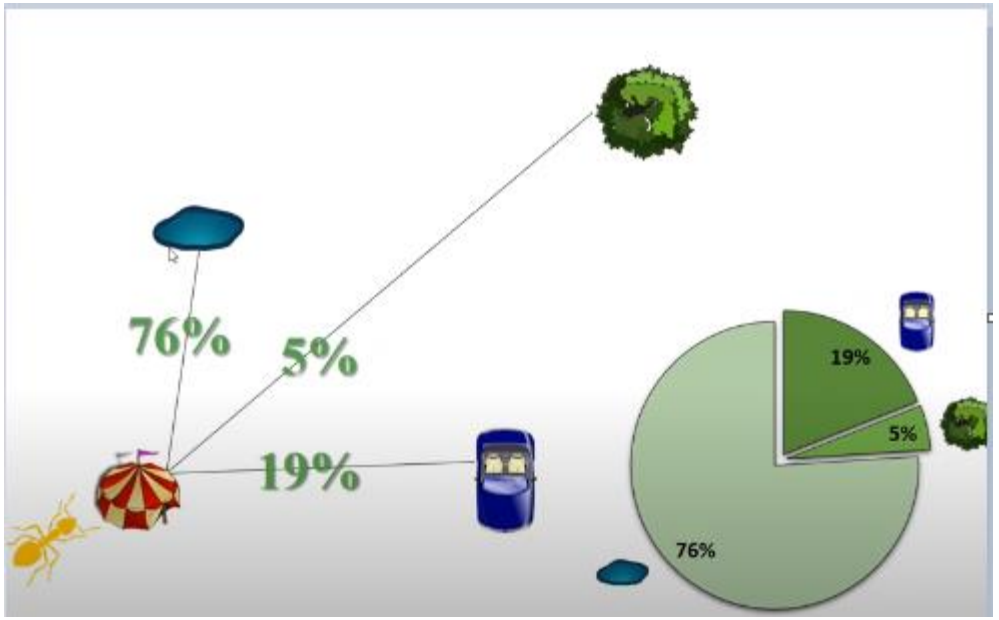
$$P = \frac{1\times\frac{1}{1}}{\left(1\times\frac{1}{1}\right)+\left(1\times\frac{1}{15}\right)+\left(1\times\frac{1}{4}\right)} = 0.7595$$

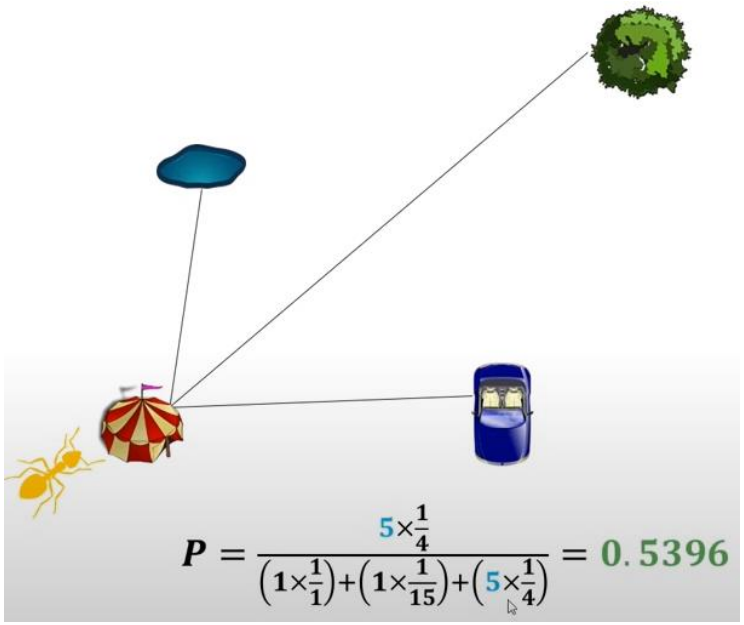




$$P = \frac{1\times\frac{1}{4}}{\left(1\times\frac{1}{1}\right)+\left(1\times\frac{1}{15}\right)+\left(1\times\frac{1}{4}\right)} = 0.1899$$

$$P = \frac{1\times\frac{1}{15}}{\left(1\times\frac{1}{1}\right)+\left(1\times\frac{1}{15}\right)+\left(1\times\frac{1}{4}\right)} = 0.0506$$

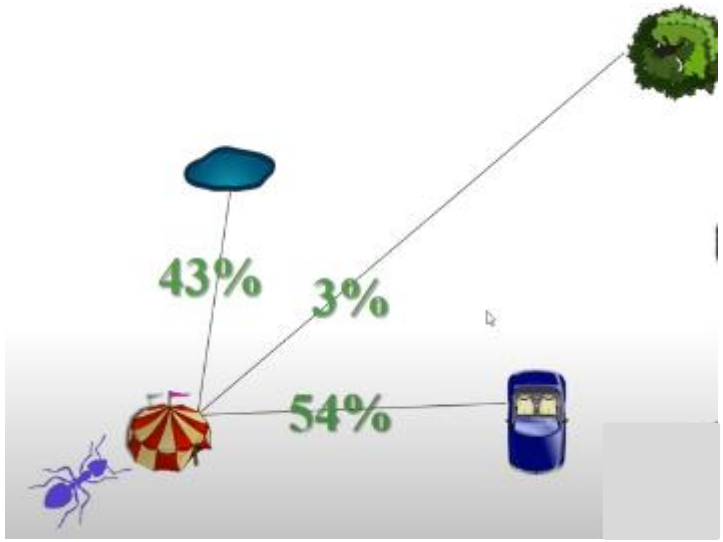


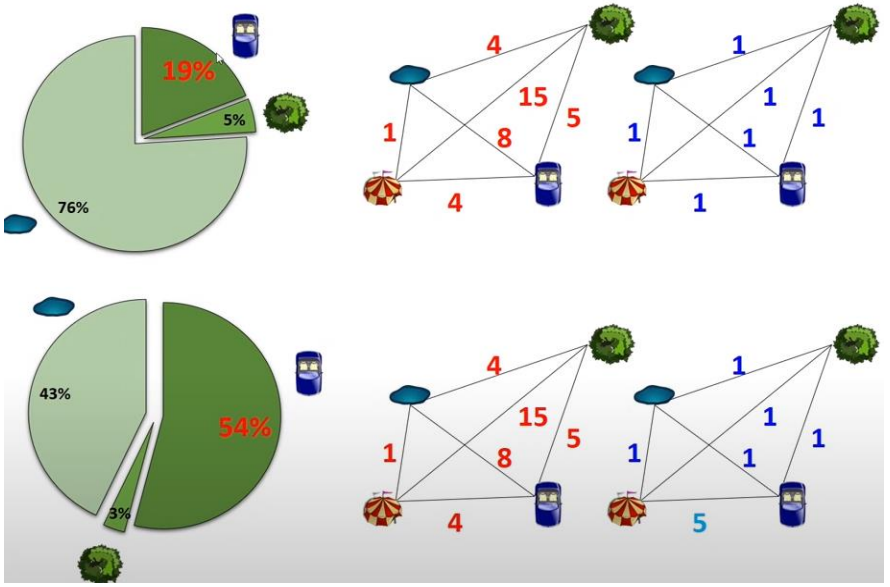


$$P = \frac{5\times\frac{1}{4}}{\left(1\times\frac{1}{1}\right)+\left(1\times\frac{1}{15}\right)+\left(5\times\frac{1}{4}\right)} = 0.5396$$

$$P = \frac{1\times\frac{1}{15}}{\left(1\times\frac{1}{1}\right)+\left(1\times\frac{1}{15}+\right)\left(5\times\frac{1}{4}\right)} = 0.0288$$

$$P = \frac{1\times\frac{1}{1}}{\left(1\times\frac{1}{1}\right)+\left(1\times\frac{1}{15}\right)+\left(5\times\frac{1}{4}\right)} = 0.4317$$





Probabilistic

0.76	0.19	0.05

Cumulative sum

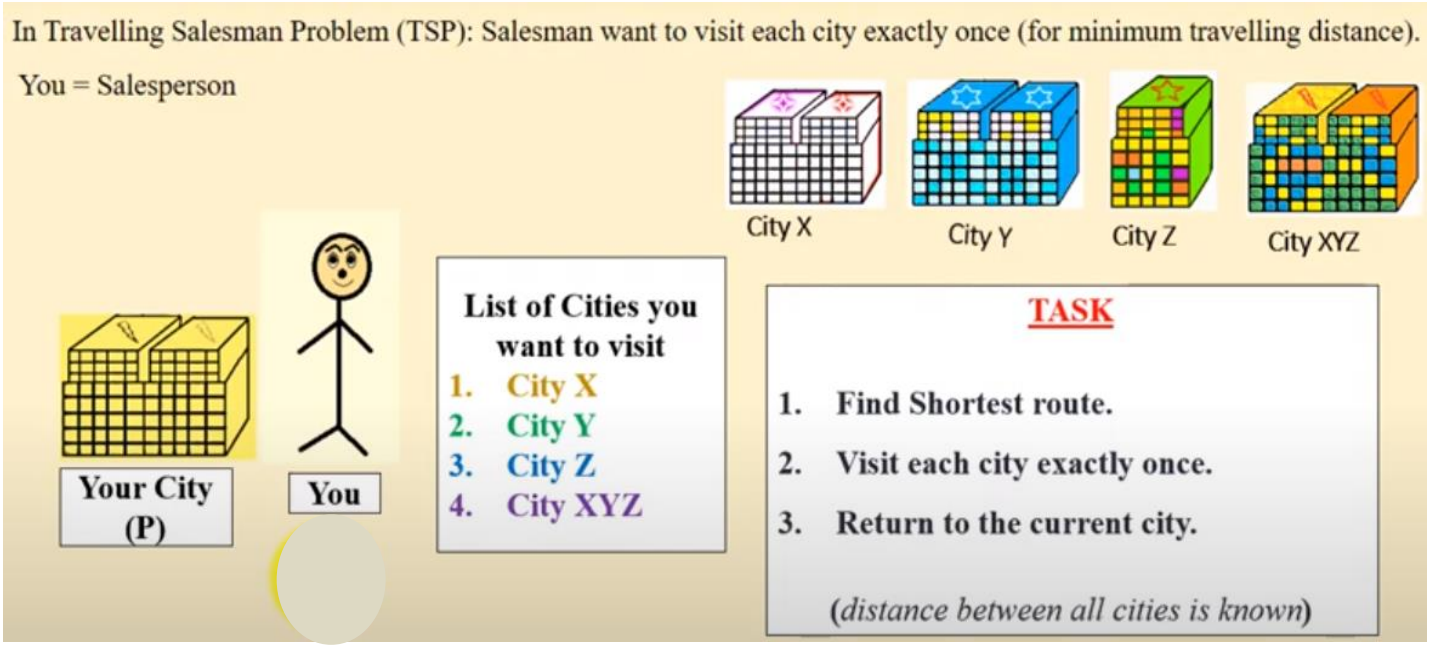
1	0.24	0.05

A random number (r) in [0,1]

$$\begin{cases} 0.24 < r \leq 1.00 \\ 0.05 < r \leq 0.24 \\ 0.00 \leq r \leq 0.05 \end{cases}$$

ANT COLONY OPTIMIZATION ALGORITHM WITH EXAMPLE

- ACO is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems.
- **Artificial ants** search for good solutions to a given optimization problem
- **Pheromone model**, that is, a set of parameters associated with graph components whose values are modified at runtime by the ants.
- We consider its application to the traveling salesman problem (TSP)



THE ANT COLONY OPTIMIZATION METAHEURISTIC

- Each instantiated decision variable $X_i = v_i^j | \mathrm{i}$ is called a *solution component* and denoted by c_{ij}

The ACO metaheuristic is:

```
Set parameters, initialize pheromone trails
SCHEDULE_ACTIVITIES
  ConstructAntSolutions
  DaemonActions    {optional}
  UpdatePheromones
END_SCHEDULE_ACTIVITIES
```

- **ConstructAntSolutions**

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \forall c_{ij} \in N(s^p)$$

- **DaemonActions**

THE ANT COLONY OPTIMIZATION METAHEURISTIC

- **UpdatePheromones**

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho \cdot \sum_{s \in S_{upd} | c_{ij} \in s} F(s)$$

$$f(s) < f(s') \Rightarrow F(s) \geq F(s'), \forall s \neq s' \in S$$

$$S_{upd} \leftarrow S_{iter}$$

$$S_{upd} \leftarrow \arg\max_{s \in S_{iter}} F(s)$$

MAIN ACO ALGORITHMS

1. Ant System

- The pheromone values are updated by *all* the ants that have completed the tour.

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_k} & \text{if } \backslash \text{ ant } k \text{ used } \backslash \text{ edge } (i,j) \text{ in } \backslash \text{ its } \backslash \text{ tour,} \\ 0 & \text{otherwise,} \end{cases}$$

- When constructing the solutions, the ants in AS traverse the construction graph and make a probabilistic decision at each vertex. The transition probability is given by:

$$p(c_{ij}|s_k^p) = \begin{cases} \frac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{c_{il}\in N(s_k^p)}\tau_{il}^{\alpha}\cdot\eta_{il}^{\beta}} & \text{if } j \in N(s_k^p), \\ 0 & \text{otherwise,} \end{cases}$$

MAIN ACO ALGORITHMS

2. Ant Colony System

- Difference between ACS and AS is the form of the decision rule used by the ants during the construction process.
- Ants in ACS use the so-called *pseudorandom proportional* rule

- The local pheromone update is performed by all ants after each construction step. Each ant applies it only to the last edge traversed:

$$\tau_{ij} = (1 - \varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0$$

- ACS offline pheromone update is performed only by the best ant, that is, only edges that were visited by the best ant are updated, according to the equation:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{best}$$

MAIN ACO ALGORITHMS

3. **MAX-MIN ant system**

- MMAS differs from AS in that

  (i) only the best ant adds pheromone trails, and

  (ii) the minimum and maximum values of the pheromone are explicitly

- The pheromone update equation takes the following form:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \Delta\tau_{ij}^{best}$$

- The pheromone values are constrained between τ_{min} and τ_{max} verifying, after they have been updated by the ants, that all pheromone values are within the imposed limits.

$$\tau_{ij}$$

- The maximum value τ_{max} may be calculated analytically provided that the optimum ant tour length is known.
- In the case of the TSP $\tau_{max} = 1/(\rho \cdot L^*)$

APPLICATIONS OF ACO AND CURRENT TRENDS

- In the domain of NP-hard combinatorial optimization problems
- In telecommunication networks
- AntNet.
- Use of ACO for the solution of dynamic, multiobjective, stochastic, continuous and mixed-variable optimization problems
- Creation of parallel implementations capable of taking advantage of the new available parallel hardware.

THANK YOU