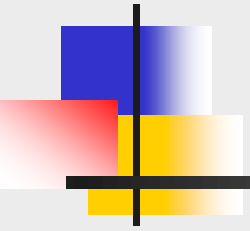


# Evolutionary Algorithm

## - Aspect & Prospect

Dr. Md. Aminul Haque Akhand
Dept. of CSE, KUET

# Evolutionary Algorithm (EA) Basics

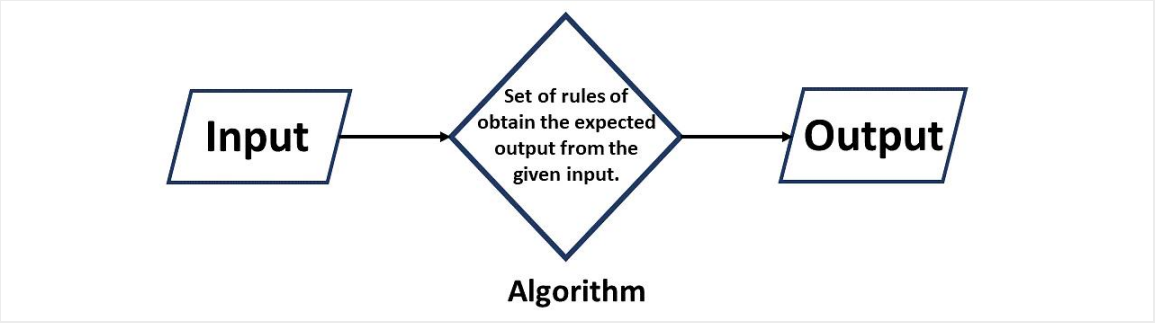
**Evolutionary**
**(Search and Optimization)**
**Algorithm**

**Evolutionary Algorithm**
**for**
**Search and Optimization**

**An Evolutionary Algorithm**
**is a subset of**
**Evolutionary Computation**

# *What is Algorithm?*

- An algorithm is a set of commands that must be followed for a computer to perform calculations or other problem-solving operations.
- According to its formal definition, an algorithm is a finite set of instructions carried out in a specific order to perform a particular task.
- It is not the entire program or code; it is simple logic to a problem represented as an informal description in the form of a flowchart or pseudocode.




https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-an-algorithm#what_is_an_algorithm

# *What is Algorithm?*

https://en.wikipedia.org/wiki/Algorithm

- In mathematics and computer science, an algorithm is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation.

- Algorithms are used as specifications for performing calculations and data processing.

- By making use of artificial intelligence, algorithms can perform automated deductions (referred to as automated reasoning) and use mathematical and logical tests to divert the code execution through various routes (referred to as automated decision-making).

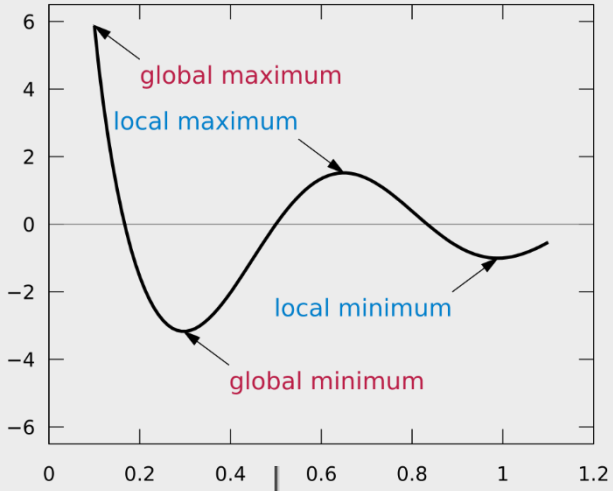
# Task: Search and Optimization

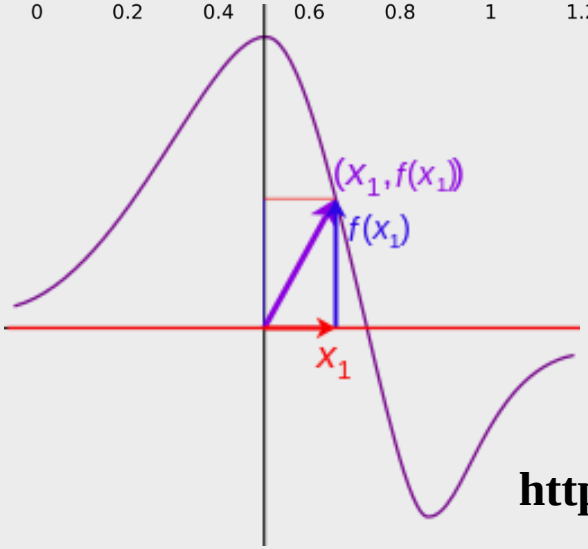
**Search (General Definition)**
**https://www.dictionary.com/browse/search**

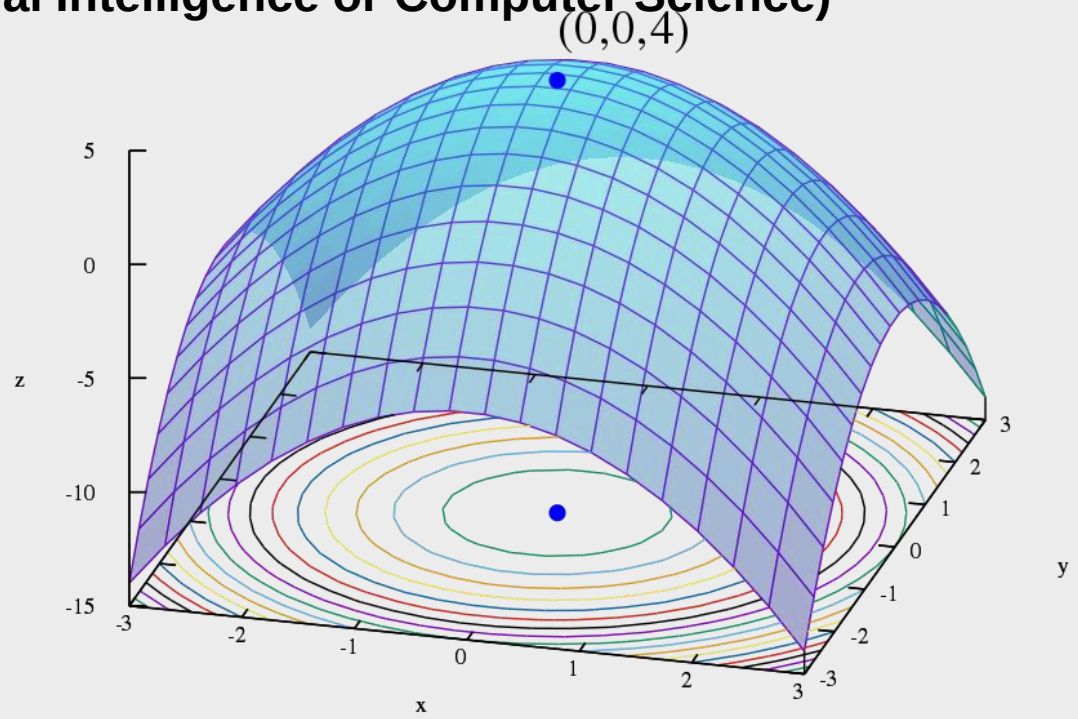
- to go or look through (a place, area, etc.) carefully in order to find something missing or lost: They searched the woods for the missing child. I searched the desk for the letter.
- to look at or examine (a person, object, etc.) carefully in order to find something concealed: The police searched the suspect for weapons.
- to explore or examine in order to discover: They searched the hills for gold.

# Task: Search and Optimization

## Search (Computational Intelligence or Computer Science)



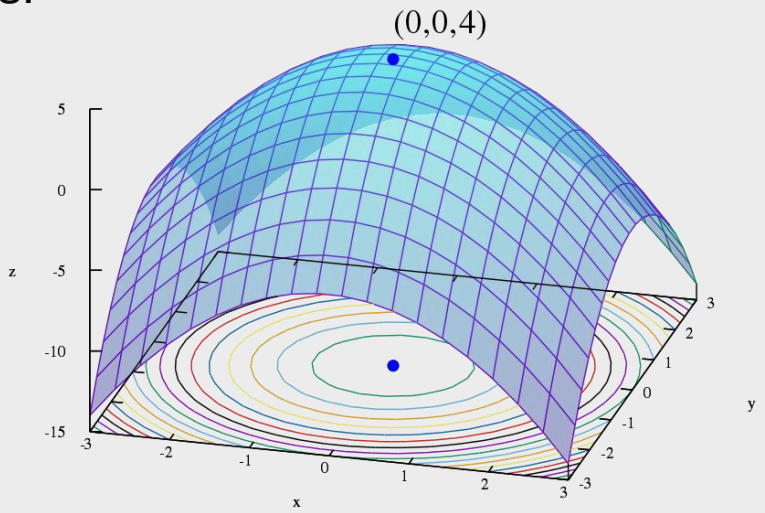

Graph of a given by $z = f(x, y) = -(x^2 + y^2) + 4$. The global maximum at $(x, y, z) = (0, 0, 4)$ is indicated by a blue dot.

**https://en.wikipedia.org/wiki/Test_functions_for_optimization**

# Task: Search and Optimization

**Optimization (General) https://www.dictionary.com/browse/optimization**

- the fact of optimizing; making the best of anything.
- the condition of being optimized.
- Mathematics: A mathematical technique for finding a maximum or minimum value of a function of several variables subject to a set of constraints, as linear programming or systems analysis.

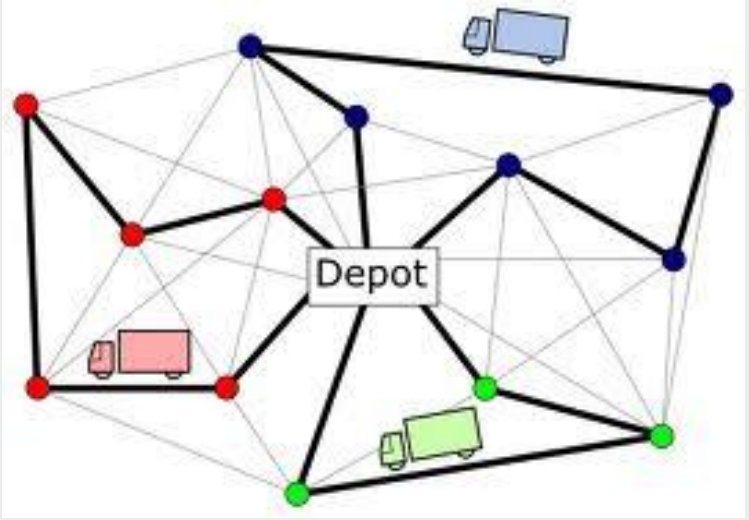


# Task: Search and Optimization

**Optimization (Computational Intelligence or Computer Science)**

**Traveling Salesman Problem (TSP)**

**Vehicle Routing Problem (VRP)**




http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

https://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/

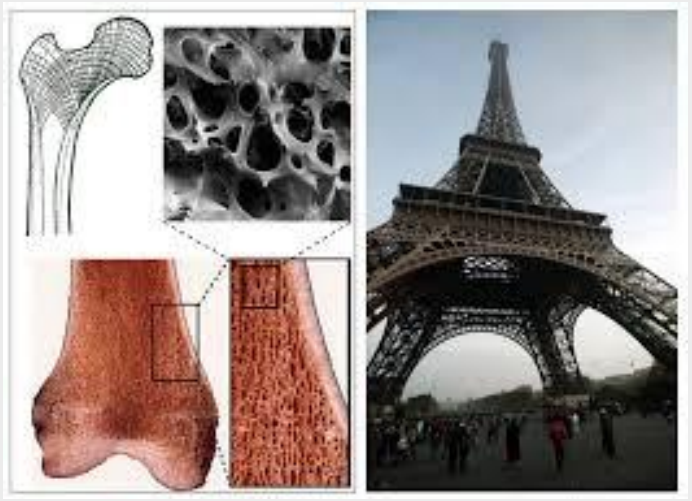
# Task: Search and Optimization

**Search and Optimization in Real-Life Scenarios**

# Solving Method: Evolutionary Approach

## Techniques taking idea from Natural Phenomena

**Biomimicry / Biomimetic: Technology is inspired by nature**



Eiffel tower designed on the femur (the human thighbone)

https://biomimicry.org/examples/

The giant water lily that inspired the London Crystal Palace

https://www.youtube.com/watch?v=HppE6ezLDqI

**Our Course Concern: Computing Techniques based on Natural Phenomena**

# Evolutionary Algorithm

https://en.wikipedia.org/wiki/Evolutionary_algorithm

In computational intelligence (CI), an evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection.

Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions (see also loss function).

Evolution of the population then takes place after the repeated application of the above operators

**Evolutionary algorithm**

Artificial development · Artificial life · Cellular evolutionary algorithm · Cultural algorithm · Differential evolution · Effective fitness · Evolutionary computation · Evolution strategy · Gaussian adaptation · Evolutionary multimodal optimization · Particle swarm optimization · Memetic algorithm · Natural evolution strategy · Neuroevolution · Promoter based genetic algorithm · Spiral optimization algorithm · Self-modifying code · Polymorphic code

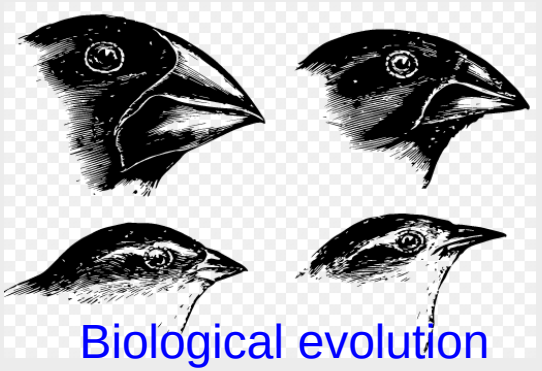
**Genetic algorithm**

Chromosome · Clonal selection algorithm · Crossover · Mutation · Genetic memory · Genetic fuzzy systems · Selection · Fly algorithm

# Evolutionary Computation

https://en.wikipedia.org/wiki/Evolutionary_computation

In computer science, evolutionary computation is a family of algorithms for global optimization inspired by biological evolution, and the subfield of artificial intelligence and soft computing studying these algorithms. In technical terms, they are a family of population-based trial and error problem solvers with a metaheuristic or stochastic optimization character.



Biological evolution

In evolutionary computation, an initial set of candidate solutions is generated and iteratively updated. Each new generation is produced by stochastically removing less desired solutions, and introducing small random changes. In biological terminology, a population of solutions is subjected to natural selection (or artificial selection) and mutation. As a result, the population will gradually evolve to increase in fitness, in this case the chosen fitness function of the algorithm.

Evolutionary computation techniques can produce highly optimized solutions in a wide range of problem settings, making them popular in computer science. Many variants and extensions exist, suited to more specific families of problems and data structures.

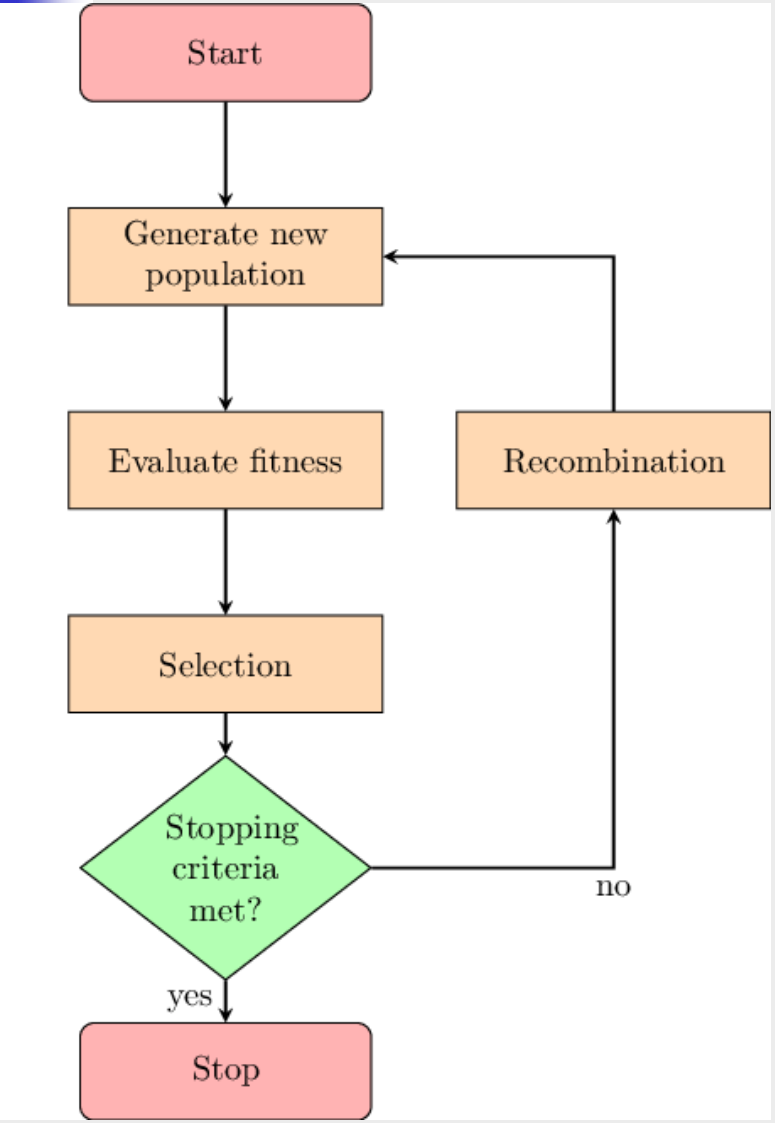
**Evolutionary algorithm**

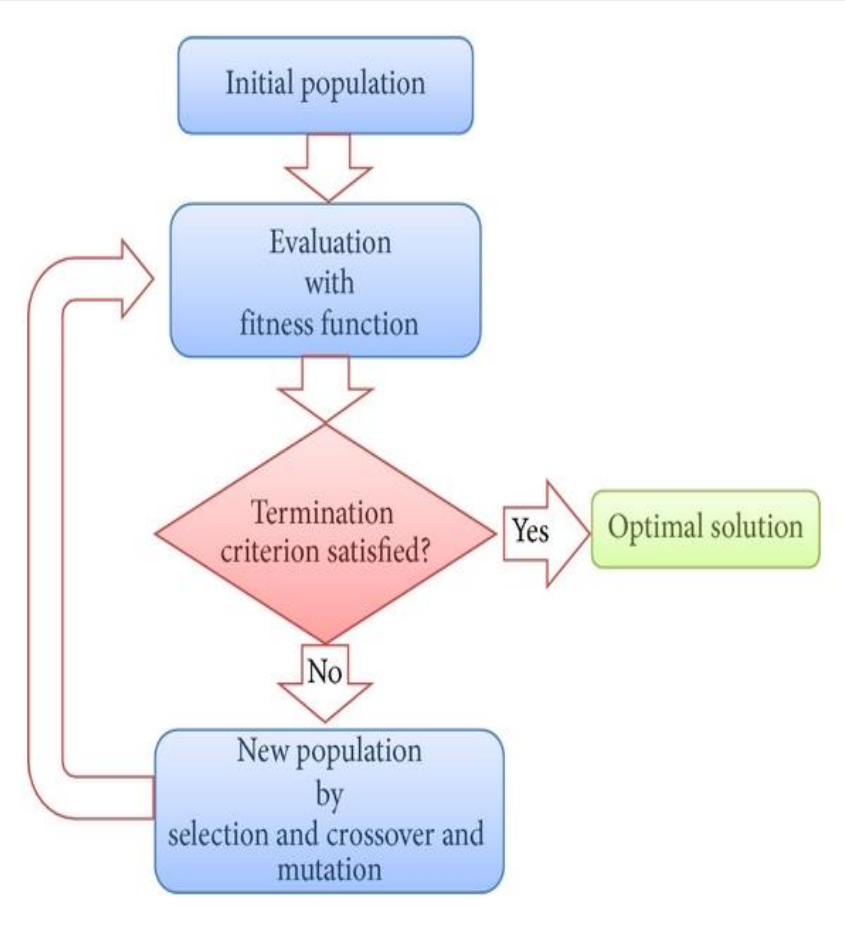
Artificial development · Artificial life · Cellular evolutionary algorithm · Cultural algorithm · Differential evolution · Effective fitness · Evolutionary computation · Evolution strategy · Gaussian adaptation · Evolutionary multimodal optimization · Particle swarm optimization · Memetic algorithm · Natural evolution strategy · Neuroevolution · Promoter based genetic algorithm · Spiral optimization algorithm · Self-modifying code · Polymorphic code

**Genetic algorithm**

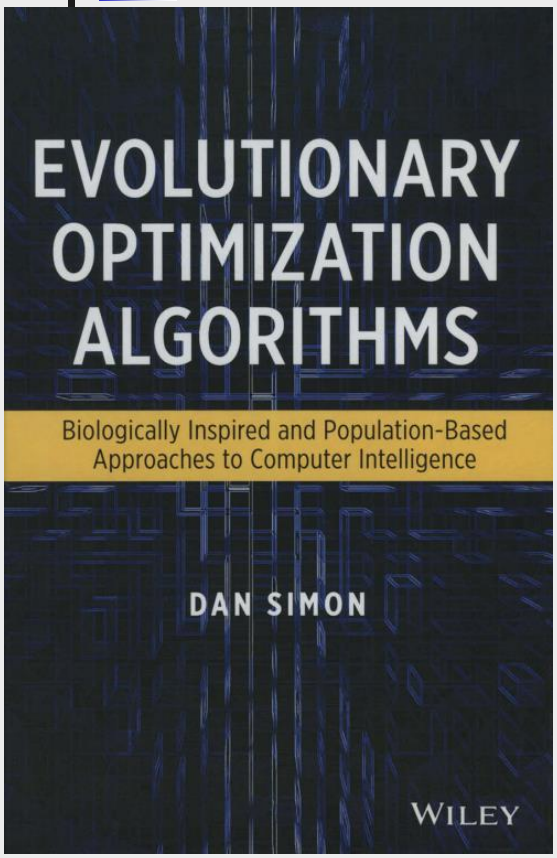
Chromosome · Clonal selection algorithm · Crossover · Mutation · Genetic memory · Genetic fuzzy systems · Selection · Fly algorithm

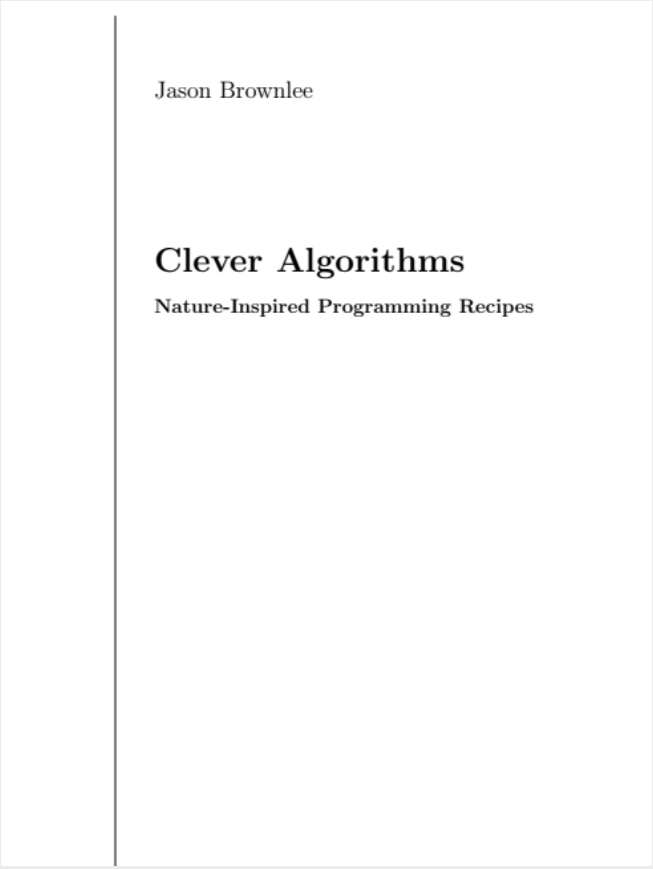
# Evolutionary Algorithm Overview

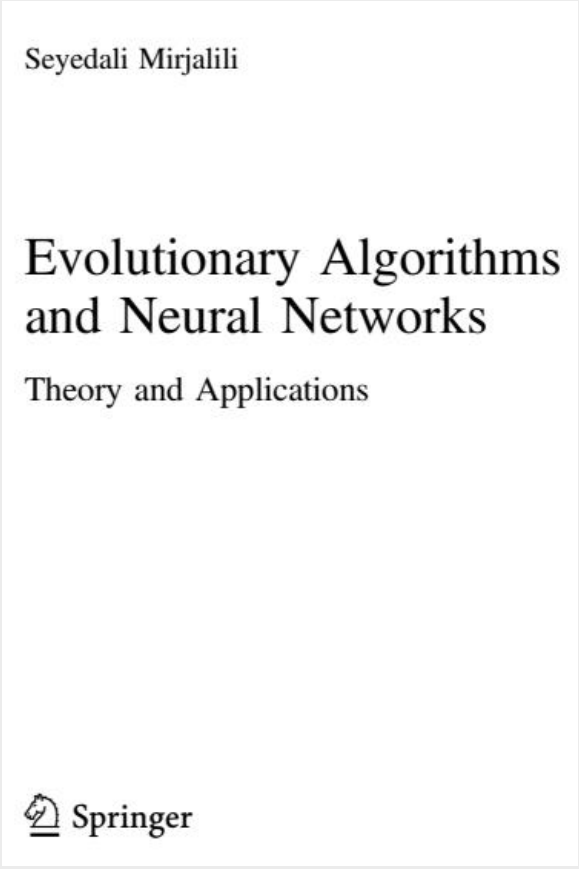


# Textbooks


EVOLUTIONARY
OPTIMIZATION
ALGORITHMS
Biologically Inspired and Population-Based
Approaches to Computer Intelligence
DAN SIMON
WILEY


Jason Brownlee
Clever Algorithms
Nature-Inspired Programming Recipes


Seyedali Mirjalili
Evolutionary Algorithms
and Neural Networks
Theory and Applications
Springer

# Research Resources

**http://www.macs.hw.ac.uk/~ml355/journals.htm**

Evolutionary Computing | Neural Computing | Natural Computing | Computational Intelligence | Optimisation and Metaheuristics | Conferences

**Evolutionary Computing**

Evolutionary Computation MIT Press, 1993-Present, Impact factor 3.600 REF →

IEEE Transactions on Evolutionary Computation IEEE Press, 1997-Present, Impact factor 5.908 REF →

Genetic Programming and Evolvable Machines Springer, 2000-Present, Impact factor 1.143 REF →

Swarm Intelligence Springer, 2007-Present, Impact factor 2.577 REF →

Evolutionary Intelligence Springer, 2008-Present REF →

Journal of Artificial Evolution and Applications Hindawi, 2008-2010 REF →

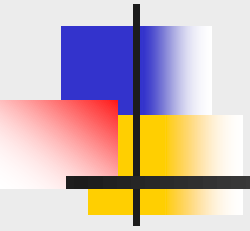
Memetic Computing Springer, 2009-Present, Impact factor 0.900 REF →

International Journal of Applied Evolutionary Computation IGI Global, 2010-Present

Swarm and Evolutionary Computation Elsevier, 2011-Present, Impact factor 2.963 REF →

International Journal of Swarm Intelligence and Evolutionary Computation OMICS group, 2012-Present

# *Open Discussion*



# Optimization

Dr. Md. Aminul Haque Akhand
Dept. of CSE, KUET

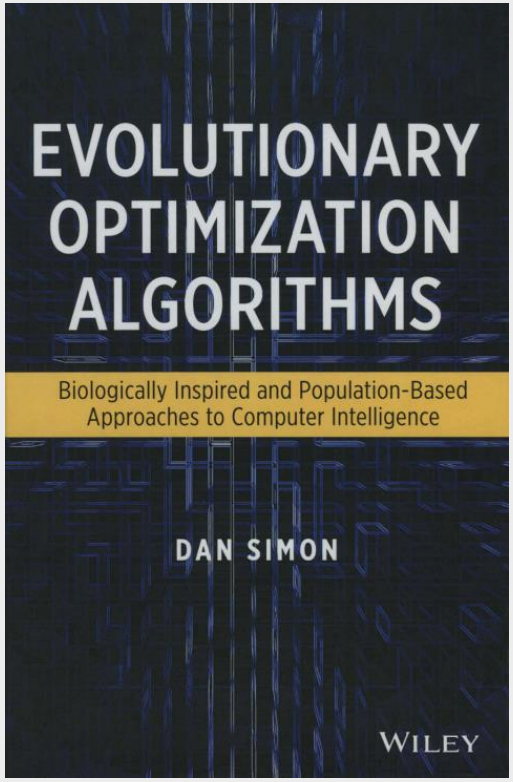
# Contents

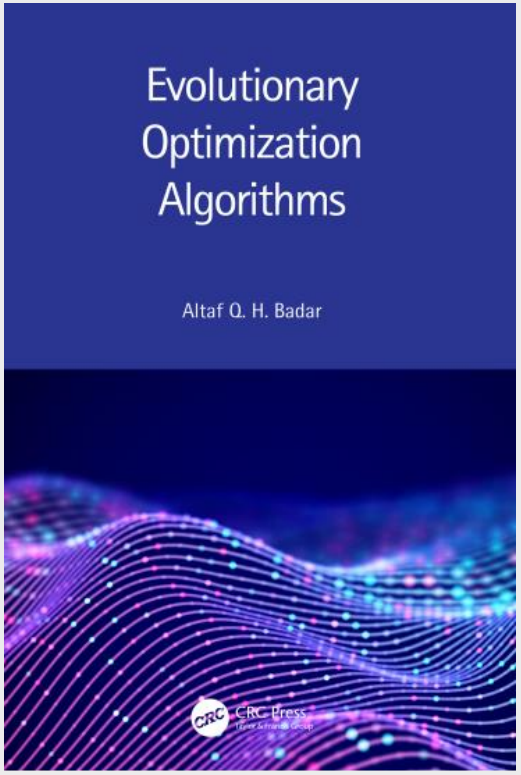
- Optimization Basics
- Unconstrained,  Multi-Objective, Multimodal, Combinatorial Optimizations
- Hill Climbing
- Intelligence

# Optimization Basics

Optimization saturates what we do and drives almost every aspect of engineering.

—Dennis Bernstein [Bernstein, 2006]



# Unconstrained Optimization

- Optimization is applicable to virtually all areas of life. Optimization algorithms can be applied to everything from aardvark breeding to zygote research.
- The possible applications of EAs are limited only by the engineer's imagination, which is why EAs have become so widely researched and applied in the past few decades.

An optimization problem can be written as a minimization problem or as a maximization problem. [Sometimes we try to minimize a function and sometimes we try to maximize a function.]

These two problems are easily converted to the other form:

$$\min_x f(x) \iff \max_x [-f(x)]$$
$$\max_x f(x) \iff \min_x [-f(x)].$$

The number of elements in *x* is called the dimension of the problem. The function *f(x)* is called the objective function, and the vector *x* is called the independent variable, or decision variable.

$$\min_x f(x) \Rightarrow f(x) \text{ is called "cost" or "objective"}$$
$$\max_x f(x) \Rightarrow f(x) \text{ is called "fitness" or "objective."}$$

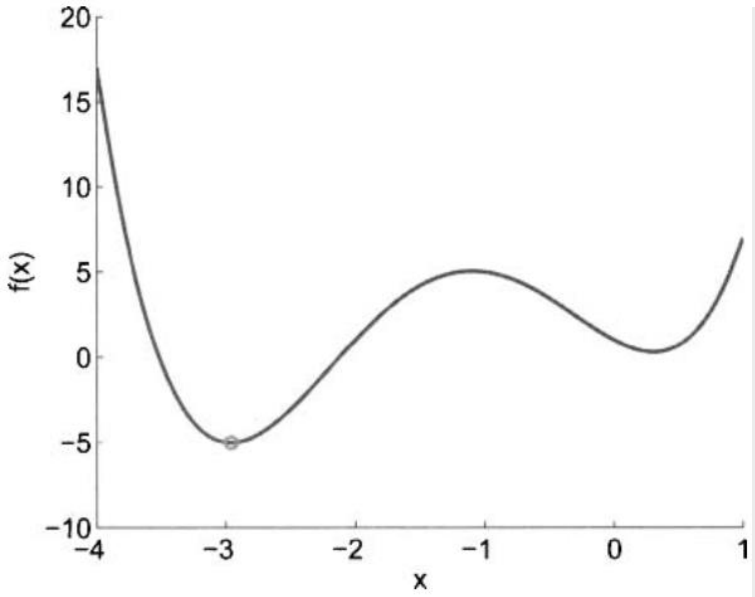
# *Unconstrained Optimization (Example)*

$$f(x, y, z) = (x - 1)^2 + (y + 2)^2 + (z - 5)^2 + 3. \qquad (2.3)$$

- *f*(x, *y, z)* is called the objective function or the cost function.
- We can change the problem to a maximization problem by defining *g(x,y,z)* = -*f(x,y,z)* and trying to maximize *g(x,y,z).* The function *g(x,y,z)* is called the objective function or the fitness function.
- The solution to the problem *min f*(*x,y,z)* is the same as the solution to the problem *max g(x,y,z)*, and is $x = 1$, $y = -2$, and $z = 5$.
- However, the optimal value of *f*(x,*y,z)* is the negative of the optimal value of *g(x,y,z).*

# Unconstrained Optimization (Example)

$$\min_x f(x), \text{ where } f(x) = x^4 + 5x^3 + 4x^2 - 4x + 1. \qquad (2.4)$$




$$f'(x) = 4x^3 + 15x^2 + 8x - 4,$$

$f'(x) = 0$ at $x = -2.96$, $x = -1.10$, and $x = 0.31$.

$$f''(x) = 12x^2 + 30x + 8 = \begin{cases} 24.33, & x = -2.96 \\ -10.48, & x = -1.10 \\ 18.45, & x = 0.31 \end{cases}$$

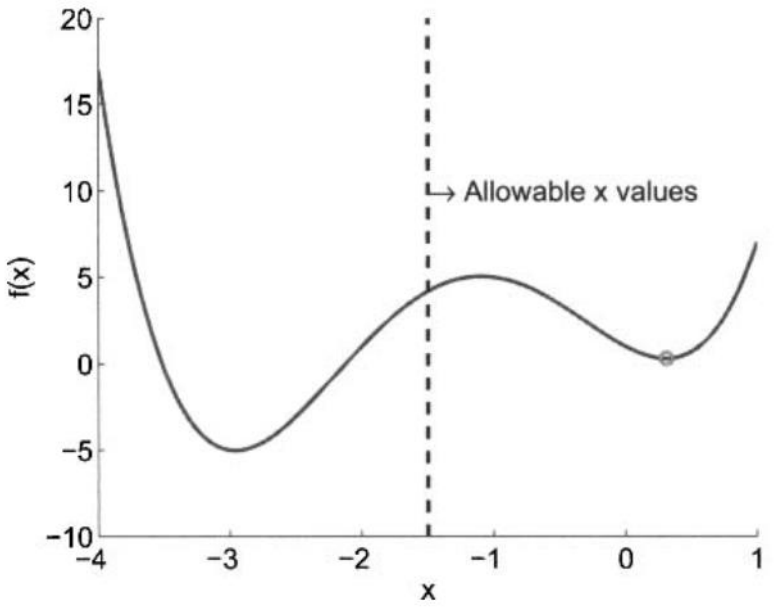
**Figure 2.1** Example 2.2: A simple minimization problem. $f(x)$ has two local minima and one global minimum, which occurs at $x = -2.96$.

Recall that the second derivative of a function at a local minimum is positive, and the second derivative of a function at a local maximum is negative. The values of $f''(x)$ at the stationary points therefore confirm that $x = -2.96$ is a local minimum, $x = -1.10$ is a local maximum, and $x = 0.31$ is another local minimum.

A local minimum $x^*$ can be defined as $f(x^*) < f(x)$ for all $x$ such that $||x - x^*|| < \epsilon$

A global minimum $x^*$ can be defined as $f(x^*) \leq f(x)$ for all $x$.

# Constrained Optimization




$$\min_x f(x) \quad \text{where} \quad f(x) = x^4 + 5x^3 + 4x^2 - 4x + 1$$
$$\text{and} \quad x \geq -1.5.$$

$$f(x) = \begin{cases} 4.19 & \text{for } x = -1.50 \\ 0.30 & \text{for } x = 0.31 \end{cases}.$$

$f'(x) = 0$ at $x = -2.96$, $x = -1.10$, and $x = 0.31$.

**Figure 2.2** Example 2.3: A simple constrained minimization problem. The constrained minimum occurs at $x = 0.31$.

# Multi-Objective Optimization

$$\min_x [f(x) \text{ and } g(x)], \quad \text{where} \quad f(x) = x^4 + 5x^3 + 4x^2 - 4x + 1$$

$$\text{and} \quad g(x) = 2(x+1)^2.$$

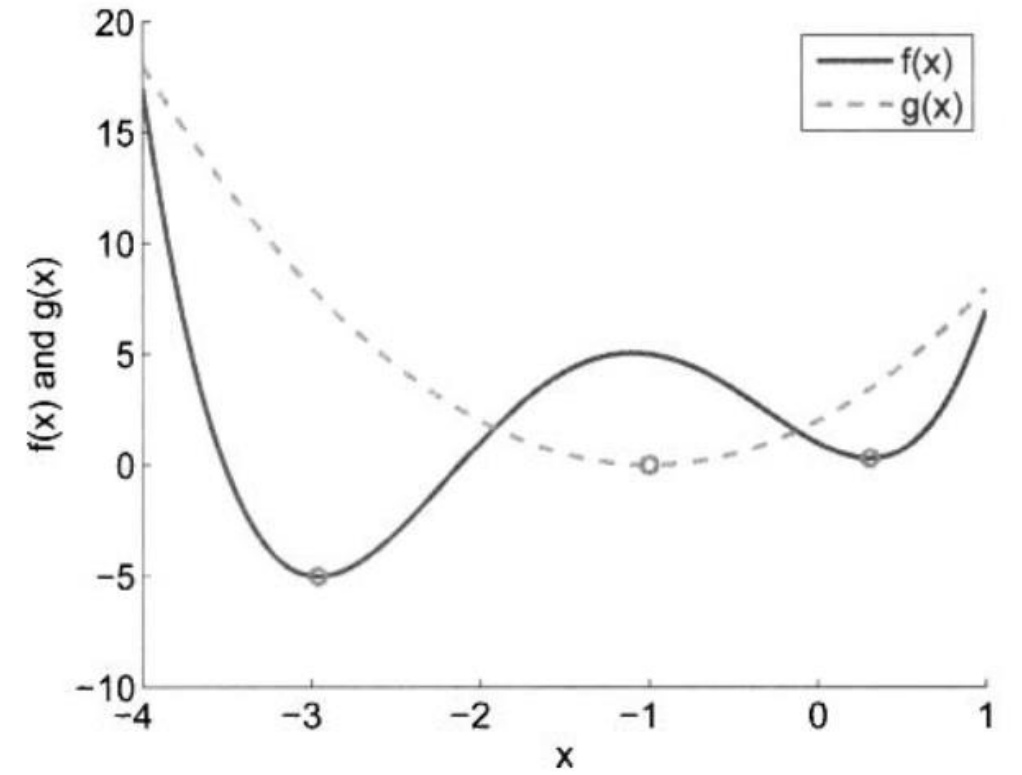
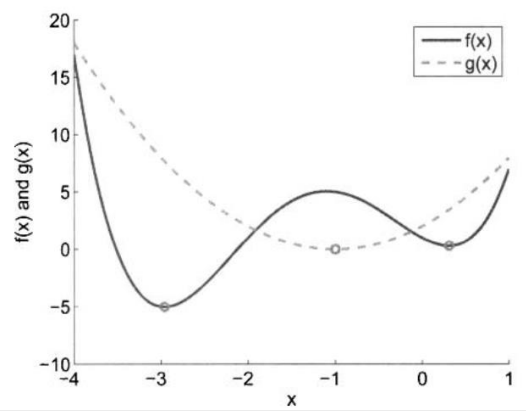



**Figure 2.3** Example 2.4: A simple multi-objective minimization problem. $f(x)$ has two minima and $g(x)$ has one minimum. The two objectives conflict.

# Multi-Objective Optimization (Cont.)

$$\min_x [f(x) \text{ and } g(x)], \quad \text{where} \quad f(x) = x^4 + 5x^3 + 4x^2 - 4x + 1$$

$$\text{and} \quad g(x) = 2(x+1)^2.$$



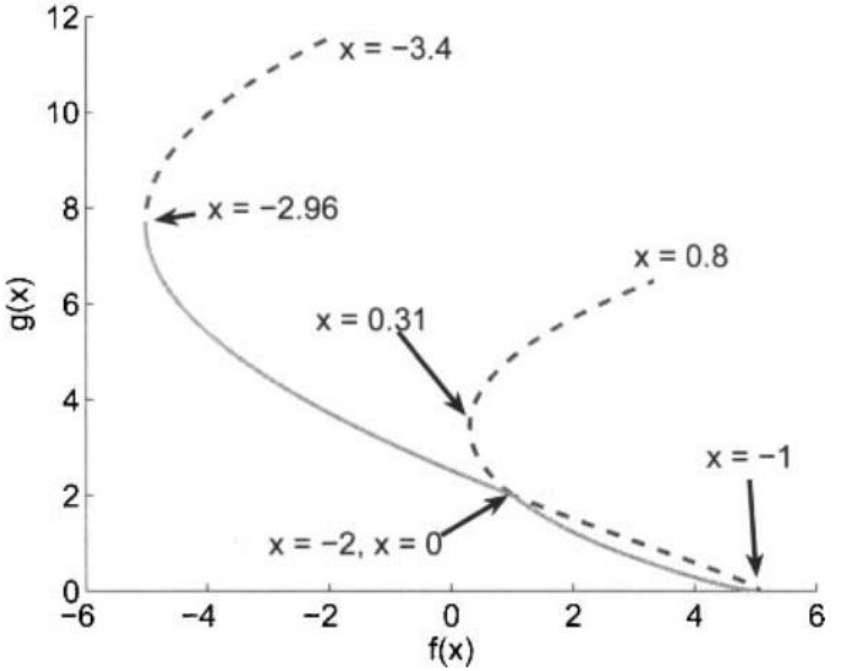




**Figure 2.4** Example 2.4: This figure shows $g(x)$ as a function of $f(x)$ for a simple multi-objective minimization problem as $x$ varies from $-3.4$ to $0.8$. The solid line is the Pareto front.

The Pareto front gives a set of reasonable choices, but any choice of *x* from the Pareto set still entails a tradeoff between the two objectives.

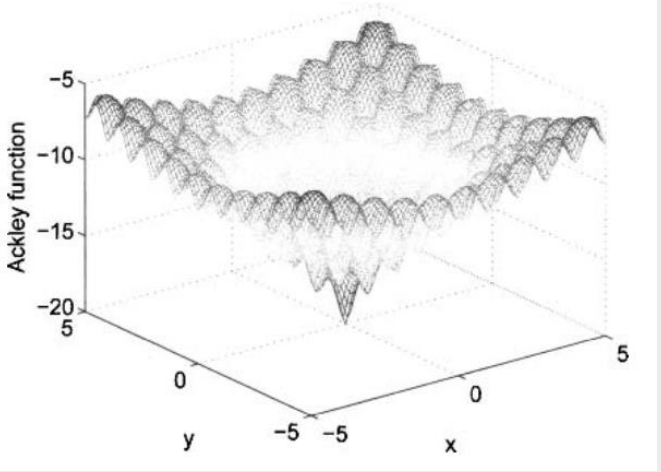
- A typical real-world optimization problem involves many more objectives, and Pareto front is difficult to obtain.
- Even if we could obtain the Pareto front, we would not be able to visualize it because of its high dimensionality.

# Multimodal Optimization

A multimodal optimization problem is a problem that has more than one local minimum.

$$\min_{x,y} f(x,y), \text{ where}$$

$$f(x,y) = e - 20\exp\left(-0.2\sqrt{\frac{x^2+y^2}{2}}\right) - \exp\left(\frac{\cos(2\pi x)+\cos(2\pi y)}{2}\right)$$



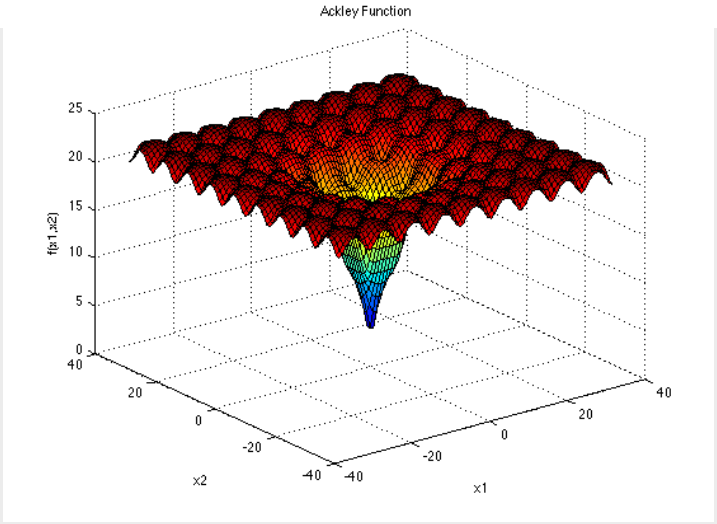




**Figure 2.5** Example 2.5: The two-dimension Ackley function.

https://www.sfu.ca/~ssurjano/ackley.html

- We were able to solve shown previous examples using graphical methods or calculus
- But many real-world problems are more like this with more independent variables, with multiple objectives, and with constraints.
- Evolutionary Algorithms (EAs) are a good choice for real-world problems.

# *Combinatorial Optimization : Traveling Salesman Problem (TSP)*

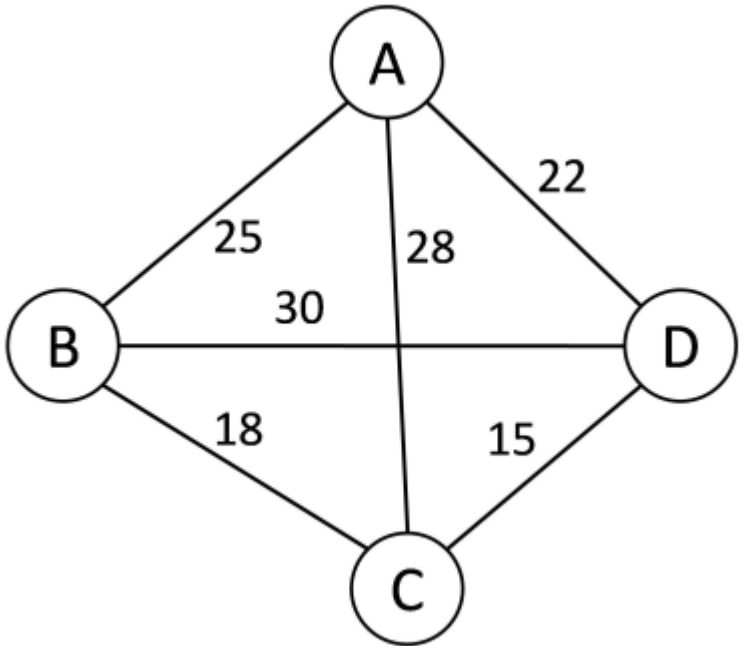



Figure 2.7: Simple Traveling Salesman Problem

There are four cities, A, B, C and D. Thus, the different possible routes would be (A as the hometown):
[A - B - C - D - A], [A - B - D - C - A],
[A - C - B - D - A], [A - C - D - B - A],
[A - D - C - B - A], [A - D - B - C - A]

If the salesman has to visit *n* cities there would be (*n*-1)! possibilities

- This number grows very rapidly, and for modest values of *n* it is not possible to calculate all possible solutions.
- Suppose the business person needs to visit one city in each of the 50 states in the USA. The number of possible solutions is 49! = $6.1 \times 10^{62}$.

# Hill Climbing

Hill climbing (HC) is a simple optimization algorithm; actually, it is a family of algorithms with many variations.

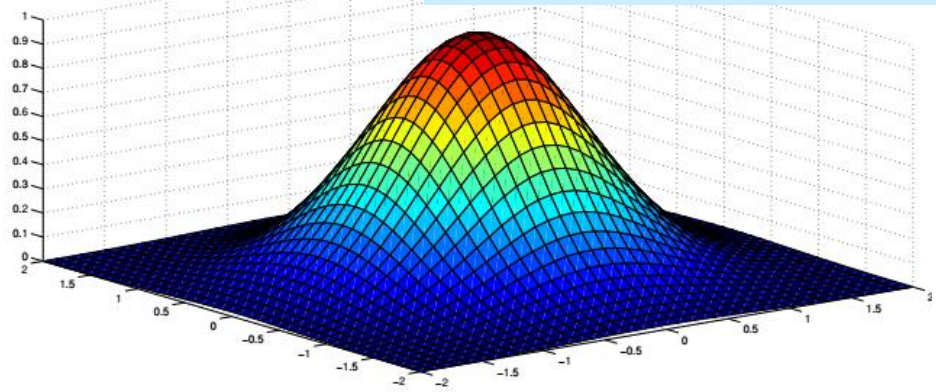


Figure 2.8: Simple Hill Climbing Problem

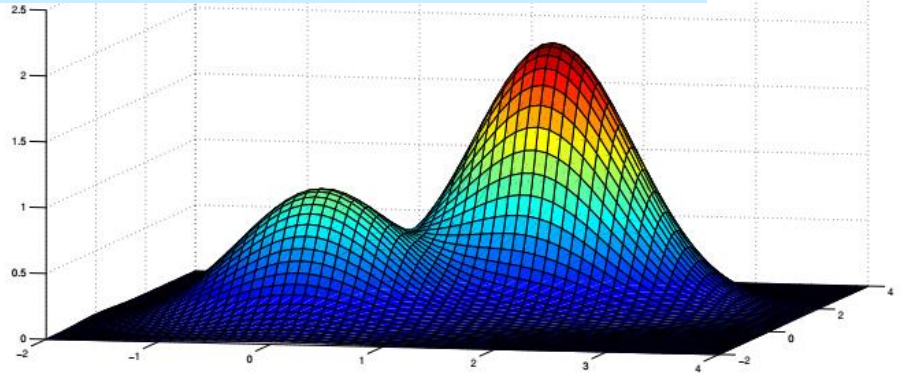


Figure 2.9: Multiple Hills in an Hill Climbing Problem

Figure 2.8 shows a simple hill-climbing surface and is the realization of the equation $e^{-(x2+y2)}$, where the value of x and y vary from -2 to 2.

Fig 2.9 is given by $e^{-(x^2+y^2)} + 2e^{-((x-1.7)^2+(y-1.7)^2)}$ in the range of $-2$ to 4.

- Some researchers consider HC to be a simple EA, while others consider it a non-evolutionary algorithm.
- HC is often a good first choice for solving a new optimization problem because it is simple, surprisingly effective
- The idea of HC is so straightforward that it must have been invented many times, and long ago, so it is difficult to determine its origin.

# Hill Climbing

The different Hill Climbing solution techniques are:

1. **Simple Hill Climbing**: In a simple hill climbing algorithm, neighboring nodes are examined one after another. The first node, which gives a better objective function value than the existing position, is selected.

2. **Steepest Ascent Hill Climbing**: In the steepest ascent hill climbing technique all the neighboring positions are evaluated. The position which gives the largest improvement in the objective function value is selected as the next node.

3. **Iterative Hill Climbing**: In this technique, the process starts with a simple random guess. Increments are done for one input variable of the random solution.

4. **Stochastic Hill Climbing**: This method chooses a neighboring position randomly and evaluates its objective function value.

5. **Random Restart/Shotgun Hill Climbing**: In the random restart hill climbing method, the hill climbing is started with a random initial position.

# Intelligence

- Computers were very good at some things that humans did poorly, like calculating the trajectory of a ballistic missile.
- But computers were (and still are) not effective at tasks that humans can do well, like recognizing a face.
- This led to attempts to mimic biological behavior in an effort to make computers better at such tasks.
- These efforts resulted in technologies like fuzzy systems, neural networks, genetic algorithms, and other EAs. EAs are therefore considered to be a part of the general category of computer intelligence.
- EAs should be *intelligent.* But what does it mean to be intelligent? Does it mean that our EAs can score high on an IQ test?
- Characteristics of Intelligence: adaptation, randomness, communication, feedback, exploration, and exploitation.

# Intelligence Characteristics

- **Adaptation:**

  Adaptation to changing environments is a feature of intelligence.

- **Randomness:**
  - ✓ Some degree of randomness is a necessary component of intelligence.
  - ✓ Too much randomness will be counterproductive.
  - ✓ So randomness is a feature of intelligence, but only within limitations.

- **Communication:**
  - ✓ Communication is a feature of intelligence.
  - ✓ Intelligence not only involves communication, but it is also emergent. A single individual cannot be intelligent.
  - ✓ Communication is required to develop intelligence, and intelligence is required to communicate.
  - ✓ Intelligence and communication form a positive feedback loop.

# Intelligence Characteristics

- **Feedback:**
  - ✓ Feedback is a fundamental characteristic of intelligence.
  - ✓ When we make mistakes, we change so that we don't repeat those mistakes. Feedback is also the basis for many natural phenomena.
  - ✓ Designed EA will have incorporate positive and negative feedback. An EA without feedback will not be very effective, but an EA with feedback has satisfied this necessary condition for intelligence.

- **Exploration and Exploitation:**
  - ✓ Exploration is the search for new ideas or new strategies.
  - ✓ Exploitation is the use of existing ideas and strategies that have proven successful in the past. Exploration is high-risk; a lot of new ideas waste time and lead to dead ends. However, exploration can also be high-return; a lot of new ideas pay off in ways that we could not have imagined.
  - ✓ Intelligence includes the proper balance of exploration and exploitation.

# Exercises

**2.1** Consider the problem $\min f(x)$, where

$$f(x) = 40 + \sum_{i=1}^{4} x_i^2 - 10\cos(2\pi x_i).$$

Note that $f(x)$ is the Rastrigin function – see Section C.1.11.

**a)** What are the independent variables of $f(x)$? What are the decision variables of $f(x)$? What are the solution features of $f(x)$?

**b)** What is the dimension of this problem?

**c)** What is the solution to this problem?

**d)** Rewrite this problem as a maximization problem.

**2.2** Consider the function $f(x) = \sin x$.

**a)** How many local minima does $f(x)$ have? What are the function values at the local minima, and what are the locally minimizing values of $x$?

**b)** How many global minima does $f(x)$ have? What are the function values at the global minima, and what are the globally minimizing values of $x$?

**2.3** Consider the function $f(x) = x^3 + 4x^2 - 4x + 1$.

**a)** How many local minima does $f(x)$ have? What are the function values at the local minima, and what are the locally minimizing values of $x$?

**b)** How many local maxima does $f(x)$ have? What are the function values at the local maxima, and what are the locally maximizing values of $x$?

**c)** How many global minima does $f(x)$ have?

**d)** How many global maxima does $f(x)$ have?

**2.4** Consider the same function as in Problem 2.3, $f(x) = x^3 + 4x^2 - 4x + 1$, but with the constraint $x \in [-5, 3]$.

**a)** How many local minima does $f(x)$ have? What are the function values at the local minima, and what are the locally minimizing values of $x$?

**b)** How many local maxima does $f(x)$ have? What are the function values at the local maxima, and what are the locally maximizing values of $x$?

**c)** How many global minima does $f(x)$ have? What is the function value at the global minimum, and what is the globally minimizing values of $x$?

**d)** How many global maxima does $f(x)$ have? What is the function value at the global maximum, and what is the globally maximizing values of $x$?

# Exercises

**2.6** How many unique closed paths exist through $N$ cities? By *unique* we mean that the starting city does not matter, and the direction of travel does not matter. For example, in a four-city problem with cities $A$, $B$, $C$, and $D$, we consider route $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$ equivalent to routes $D \rightarrow C \rightarrow B \rightarrow A \rightarrow D$ and $B \rightarrow C \rightarrow D \rightarrow A \rightarrow B$.

**2.7** Consider the closed TSP with the cities in Table 2.2.

| City | $x$ | $y$ |
|---|---|---|
| A | 5 | 9 |
| B | 9 | 8 |
| C | −6 | −8 |
| D | 9 | −2 |
| E | −5 | 9 |
| F | 4 | −7 |
| G | −9 | 1 |

**Table 2.2** TSP coordinates of cities for Problem 2.7.

**a)** How many closed routes exist through these seven cities?

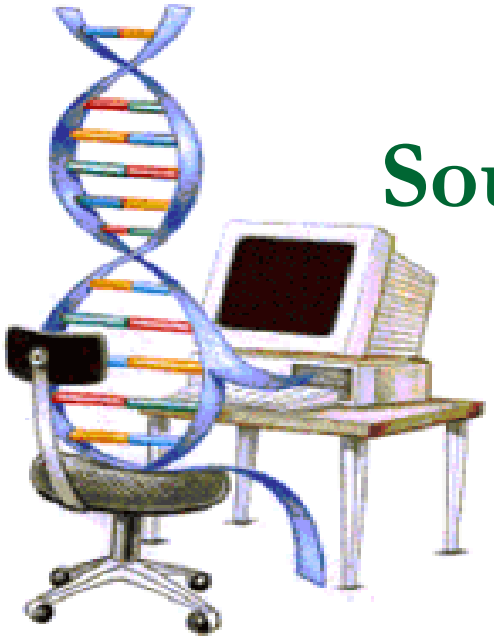
**b)** Is it easy to see the solution by looking at the coordinates in Table 2.2?

**c)** Plot the coordinates. Is it easy to see the solution from the plot? What is the optimal solution? This problem shows that looking at a problem in a different way might help us find a solution.

# *Open Discussion*

# Genetic Algorithms

Prof. Dr. Md. Aminul Haque Akhand
Dept. of CSE, KUET



## Source

1. **Presentation of Dr. Chhavi Kashyap**
2. **Presentation of Dr. Son Kuswadi**
3. **Others**

# Overview

- Introduction To Genetic Algorithms (GAs)
- GA Operators and Parameters
- Application of Genetic Algorithms
- Summary

# References


- D. E. Goldberg, 'Genetic Algorithm In Search, Optimization And Machine Learning', New York: Addison – Wesley (1989)
- John H. Holland 'Genetic Algorithms', Scientific American Journal, July 1992.
- Kalyanmoy Deb, 'An Introduction To Genetic Algorithms', Sadhana, Vol. 24 Parts 4 And 5.
- D. Whitley, et al, 'Traveling Salesman And Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination', Handbook Of Genetic Algorithms, New York
- Jason Brownlee, "Clever Algorithms - Nature-Inspired Programming Recipes" 2011.


**WEBSITES**

www.iitk.ac.in/kangal www.genetic-programming.com

# Introduction To Genetic Algorithms (GAs)

# History Of Genetic Algorithms

- "Evolutionary Computing" was introduced in the 1960s by **I. Rechenberg**.
- John Holland wrote the first book on Genetic Algorithms '*Adaptation in Natural and Artificial Systems*' in 1975.
- In 1992 **John Koza** used genetic algorithm to evolve programs to perform certain tasks. He called his method "Genetic Programming".

"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change."

-------- Charles Darwin

(12/02/1809 – 19/04/1882)

# Basic Idea Of Principle Of Natural Selection

***"Select The Best, Discard The Rest"***

# Darwin's Principle Of Natural Selection

- IF there are organisms that reproduce, and
- IF offsprings inherit traits from their progenitors, and
- IF there is variability of traits, and
- IF the environment cannot support all members of a growing population,
- THEN those members of the population with less-adaptive traits (determined by the environment) will die out, and
- THEN those members with more-adaptive traits (determined by the environment) will thrive

The result is the evolution of species.

# An Example of Natural Selection

- **Giraffes have long necks**.

Giraffes with slightly longer necks could feed on leaves of higher branches when all lower ones had been eaten off.

→ They had a better chance of survival.

→ Favorable characteristic propagated through generations of giraffes.

→ Now, evolved species has long necks.

NOTE: Longer necks may have been a deviant characteristic (mutation) initially but since it was favorable, was propagated over generations. Now an established trait.

So, some mutations are beneficial.

# Evolution Through Natural Selection

Initial Population Of Animals

↓

Struggle For Existence-Survival Of the Fittest

↓

Surviving Individuals Reproduce, Propagate Favorable Characteristics

↓

**Millions Of Years**

↓

Evolved Species

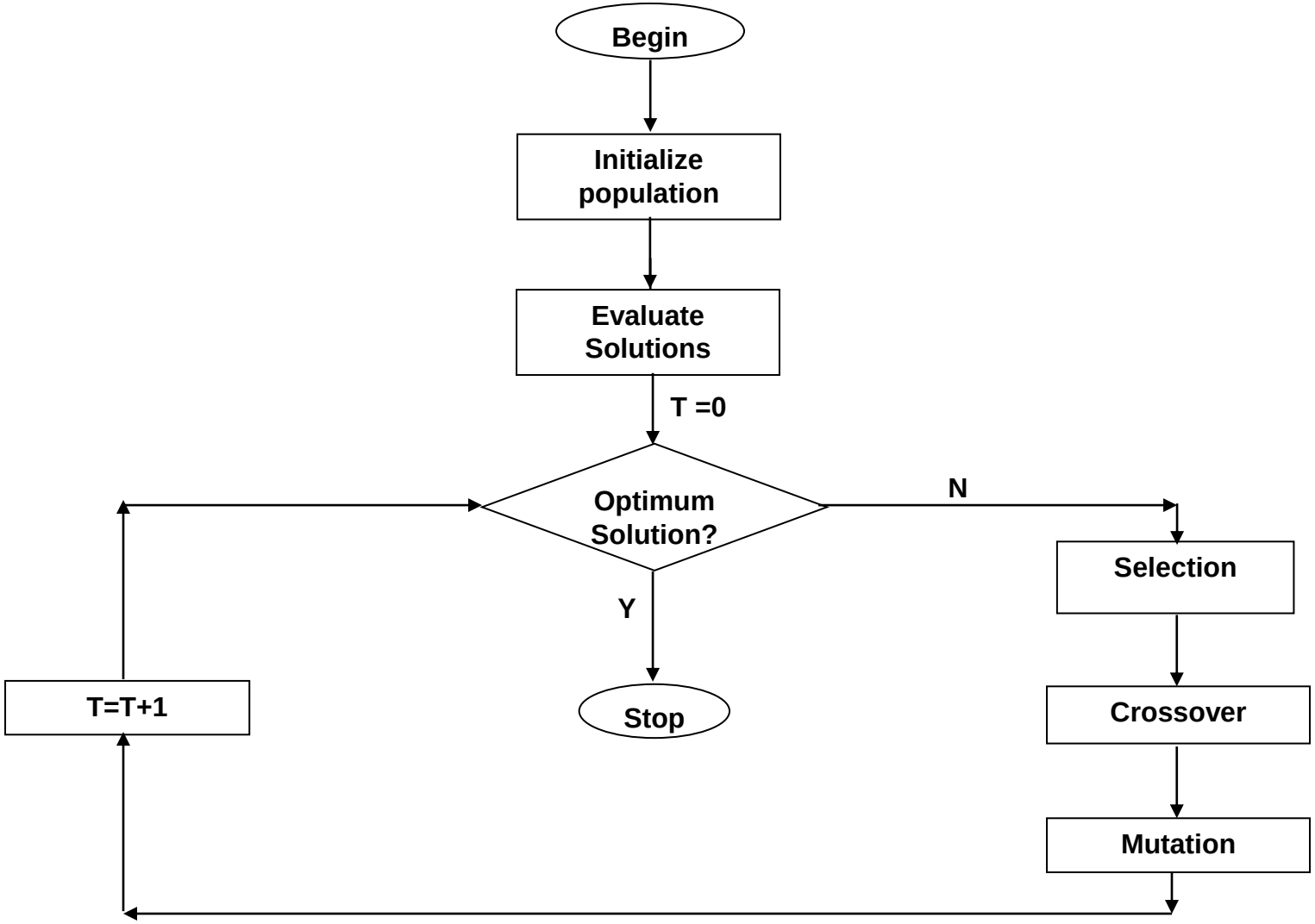
**(Favorable Characteristic Now A Trait Of Species)**

# What Are Genetic Algorithms (GAs)?

Genetic Algorithms are *search* and *optimization* techniques based on Darwin's Principle of *Natural Selection*.

Genetic Algorithms Implement Optimization Strategies By Simulating Evolution Of Species Through Natural Selection.

# Working Mechanism Of GAs



# Simple Genetic Algorithm

```
Simple_Genetic_Algorithm()
      {
      Initialize the Population;
      Calculate Fitness Function;

      While(Fitness Value != Optimal Value)
            {
            Selection;//Natural Selection, Survival
  Of Fittest
            Crossover;//Reproduction, Propagate
  favorable characteristics

            Mutation;//Mutation
            Calculate Fitness Function;
            }
      }
```

# Nature to Computer Mapping

| Nature | Computer |
|---|---|
| Population | Set of solutions. |
| Individual | Solution to a problem. |
| Fitness | Quality of a solution. |
| Chromosome | Encoding for a Solution. |
| Gene | Part of the encoding of a solution. |
| Reproduction | Crossover |

# GA Operators and Parameters

# Encoding

*The process of representing the solution in the form of a **string** that conveys the necessary information.*

- Just as in a chromosome, each gene controls a particular characteristic of the individual, similarly, each bit in the string represents a characteristic of the solution.

# Encoding Methods

- **Binary Encoding –** Most common method of encoding. Chromosomes are strings of 1s and 0s and each position in the chromosome represents a particular characteristic of the problem.

| | |
|---|---|
| Chromosome A | 10110010110011100101 |
| Chromosome B | 11111110000000011111 |

# Encoding Methods (contd.)

- **Permutation Encoding –** Useful in ordering problems such as the Traveling Salesman Problem (TSP). Example. In TSP, every chromosome is a string of numbers, each of which represents a city to be visited.

| | |
|---|---|
| Chromosome A | 1 5 3 2 6 4 7 9 8 |
| Chromosome B | 8 5 6 7 2 3 1 4 9 |

# Encoding Methods (contd.)

- **Value Encoding –** Used in problems where complicated values, such as real numbers, are used and where binary encoding would not suffice.

  Good for some problems, but *often necessary to develop some specific crossover and mutation techniques for these chromosomes.*

| | |
|---|---|
| Chromosome A | 1.235 5.323 0.454 2.321 2.454 |
| Chromosome B | (left), (back), (left), (right), (forward) |

# Fitness Function

*A fitness function quantifies the optimality of a solution (chromosome) so that that particular solution may be ranked against all the other solutions.*

- A fitness value is assigned to each solution depending on how close it actually is to solving the problem.
- Ideal fitness function correlates closely to goal + quickly computable.
- Example. In TSP, f(x) is sum of distances between the cities in solution. The lesser the value, the fitter the solution is.

# Recombination

*The process that determines which solutions are to be preserved and allowed to reproduce and which ones deserve to die out.*

- The primary objective of the recombination operator is to emphasize the good solutions and eliminate the bad solutions in a population, while keeping the population size constant.
- “Selects The Best, Discards The Rest”.
- “Recombination” is different from “Reproduction”.

# Recombination

- Identify the good solutions in a population.
- Make multiple copies of the good solutions.
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population.

# Roulette Wheel Selection

- Each current string in the population has a slot assigned to it which is in proportion to it's fitness.

- We spin the weighted *roulette wheel* thus defined *n* times (where *n* is the total number of solutions).

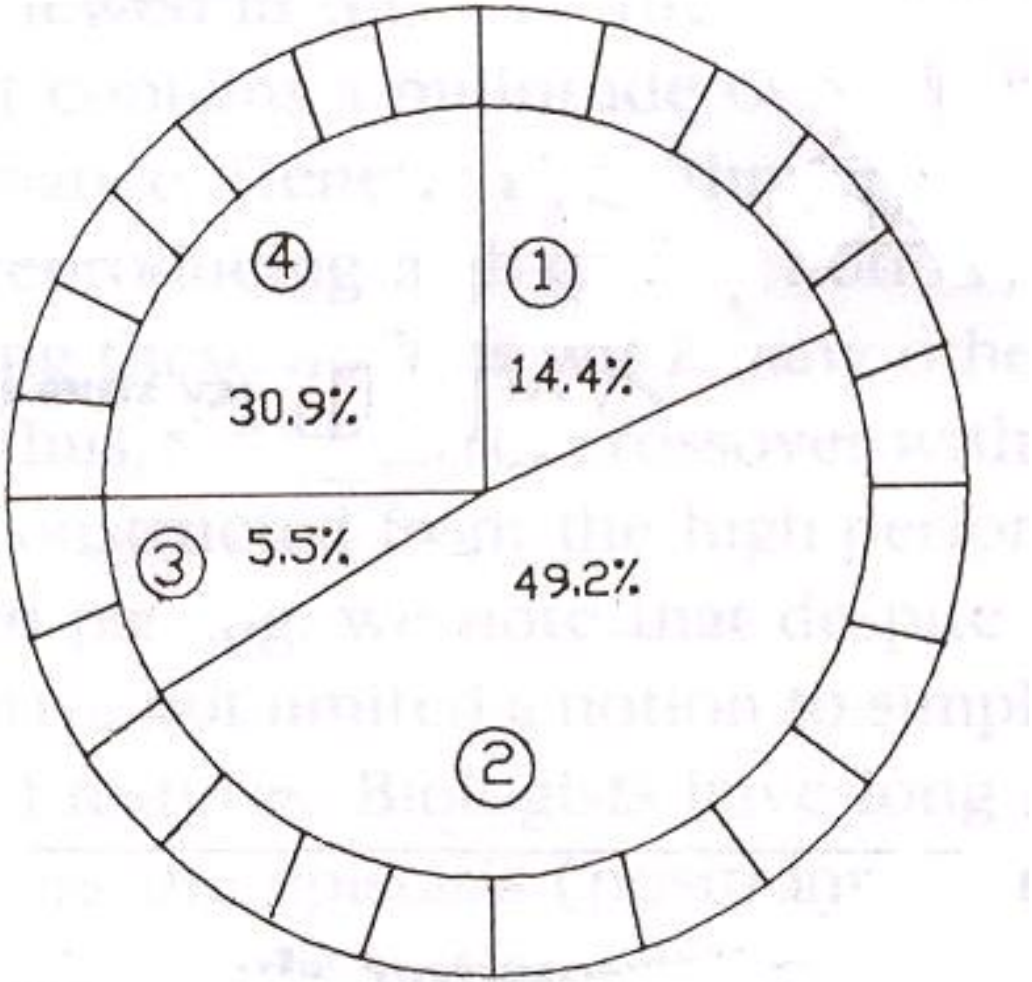
- Each time Roulette Wheel stops, the string corresponding to that slot is created.

Strings that are fitter are assigned a larger slot and hence have a better chance of appearing in the new population.

# Example Of Roulette Wheel Selection

| No. | String | Fitness | % Of Total |
|---|---|---|---|
| 1 | 01101 | 169 | 14.4 |
| 2 | 11000 | 576 | 49.2 |
| 3 | 01000 | 64 | 5.5 |
| 4 | 10011 | 361 | 30.9 |
| **Total** | | 1170 | 100.0 |

# Roulette Wheel For Example



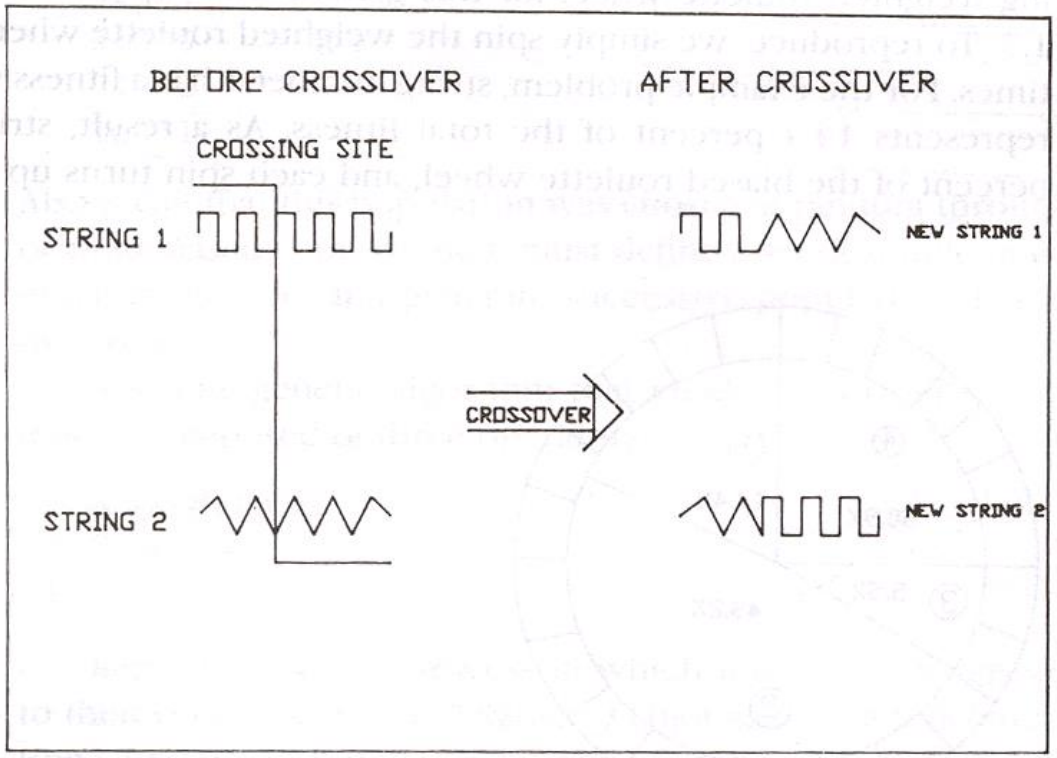
# Crossover

*It is the process in which two chromosomes (strings) combine their genetic material (bits) to produce a new offspring which possesses both their characteristics.*

- *Two strings are picked from the mating pool at random to cross over.*
- *The method chosen depends on the Encoding Method.*

# Crossover Methods

- **Single Point Crossover-** A random point is chosen on the individual chromosomes (strings) and the genetic material is exchanged at this point.



# Crossover Methods (contd.)

- **Single Point Crossover**

| | |
|---|---|
| Chromosome1 | 11011 \| 00100110110 |
| Chromosome 2 | 11011 \| 11000011110 |
| Offspring 1 | 11011 \| 11000011110 |
| Offspring 2 | 11011 \| 00100110110 |

# Crossover Methods (contd.)

- **Two-Point Crossover-** Two random points are chosen on the individual chromosomes (strings) and the genetic material is exchanged at these points.

| | |
|---|---|
| **Chromosome1** | **11011 \| 00100 \| 110110** |
| **Chromosome 2** | **10101 \| 11000 \| 011110** |
| **Offspring 1** | **10101 \| 00100 \| 011110** |
| **Offspring 2** | **11011 \| 11000 \| 110110** |

**NOTE: These chromosomes are different from the last example.**

# Crossover Methods (contd.)

- **Uniform Crossover-** Each gene (bit) is selected randomly from one of the corresponding genes of the parent chromosomes.

| | |
|---|---|
| **Chromosome1** | **11011 \| 00100 \| 110110** |
| **Chromosome 2** | **10101 \| 11000 \| 011110** |
| **Offspring** | **10111 \| 00000 \| 110110** |

**NOTE: Uniform Crossover yields ONLY 1 offspring.**

# Crossover (contd.)

- Crossover between 2 good solutions **MAY NOT ALWAYS** yield a better or as good a solution.
- Since parents are good, probability of the child being good is high.
- If offspring is not good (poor solution), it will be removed in the next iteration during “Selection”.

# Elitism

*Elitism is a method which copies the best chromosome to the new offspring population before crossover and mutation.*

- When creating a new population by crossover or mutation the best chromosome might be lost.
- Forces GAs to retain some number of the best individuals at each generation.
- Has been found that elitism significantly improves performance.

# Mutation

*It is the process by which a string is deliberately changed so as to maintain diversity in the population set.*

We saw in the giraffes' example, that mutations could be beneficial.

***Mutation Probability-*** determines how often the parts of a chromosome will be mutated.

# Example Of Mutation

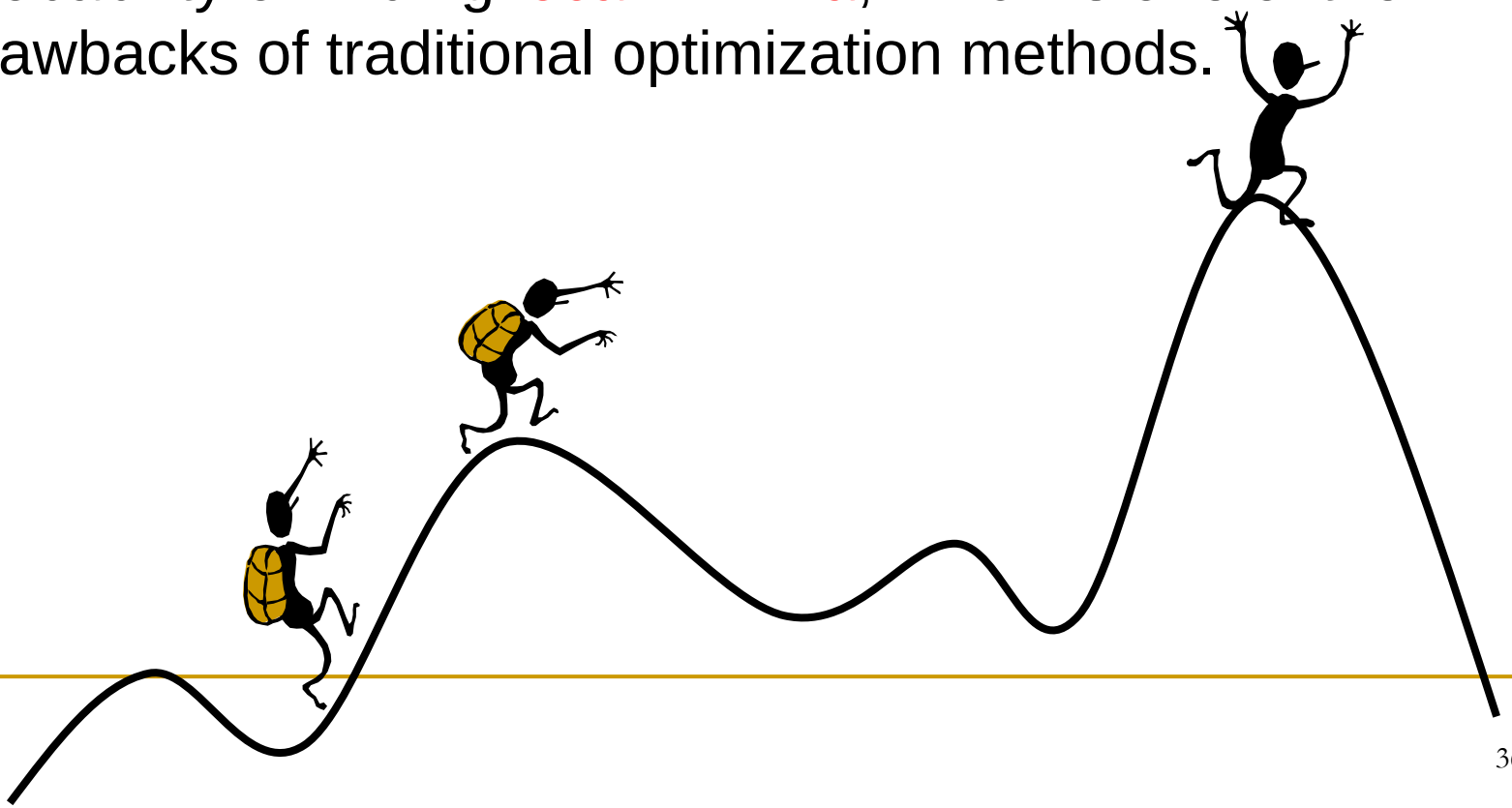
- For chromosomes using Binary Encoding, randomly selected bits are inverted.

| | |
|---|---|
| **Offspring** | **11011 00100 10110** |
| **Mutated Offspring** | **11010 00100 10110** |

**NOTE: The number of bits to be inverted depends on the Mutation Probability.**

# Advantages Of GAs

- **Global Search Methods:** GAs search for the function optimum starting from a *population of points* of the function domain, not a single one. This characteristic suggests that GAs are global search methods. They can, in fact, climb many peaks in parallel, reducing the probability of finding local minima, which is one of the drawbacks of traditional optimization methods.



# Advantages of GAs (contd.)

- **Blind Search Methods:** GAs only use the information about the *objective function*. They do not require knowledge of the first derivative or any other auxiliary information, allowing a number of problems to be solved without the need to formulate restrictive assumptions. For this reason, GAs are often called blind search methods.

# Advantages of GAs (contd.)

- **GAs use probabilistic transition rules** during iterations, unlike the traditional methods that use fixed transition rules.

  This makes them more robust and applicable to a large range of problems.

# Advantages of GAs (contd.)

- **GAs can be easily used in *parallel machines*-** Since in real-world design optimization problems, most computational time is spent in evaluating a solution, with multiple processors all solutions in a population can be evaluated in a distributed manner. This reduces the overall computational time substantially.

# Simple Example (Goldberg98)

- **Simple problem: max $x^2$ over {0,1,…,31}**
- GA approach:
  - Representation: binary code, e.g. 01101 ↔ 13
  - Population size: 4
  - 1-point xover, bitwise mutation
  - Roulette wheel selection
  - Random initialization
- We show one generational cycle done by hand

# Selection

| String no. | Initial population | $x$ Value | Fitness $f(x) = x^2$ | $Prob_i$ | Expected count | Actual count |
|---|---|---|---|---|---|---|
| 1 | 0 1 1 0 1 | 13 | 169 | 0.14 | 0.58 | 1 |
| 2 | 1 1 0 0 0 | 24 | 576 | 0.49 | 1.97 | 2 |
| 3 | 0 1 0 0 0 | 8 | 64 | 0.06 | 0.22 | 0 |
| 4 | 1 0 0 1 1 | 19 | 361 | 0.31 | 1.23 | 1 |
| Sum | | | 1170 | 1.00 | 4.00 | 4 |
| Average | | | 293 | 0.25 | 1.00 | 1 |
| Max | | | 576 | 0.49 | 1.97 | 2 |

# Crossover

| String no. | Mating pool | Crossover point | Offspring after xover | $x$ Value | Fitness $f(x) = x^2$ |
|---|---|---|---|---|---|
| 1 | 0 1 1 0 \| 1 | 4 | 0 1 1 0 0 | 12 | 144 |
| 2 | 1 1 0 0 \| 0 | 4 | 1 1 0 0 1 | 25 | 625 |
| 2 | 1 1 \| 0 0 0 | 2 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 \| 0 1 1 | 2 | 1 0 0 0 0 | 16 | 256 |
| Sum | | | | | 1754 |
| Average | | | | | 439 |
| Max | | | | | 729 |

# Mutation

| String no. | Offspring after xover | Offspring after mutation | $x$ Value | Fitness $f(x) = x^2$ |
|---|---|---|---|---|
| 1 | 0 1 1 0 0 | 1 1 1 0 0 | 26 | 676 |
| 2 | 1 1 0 0 1 | 1 1 0 0 1 | 25 | 625 |
| 2 | 1 1 0 1 1 | 1 1 0 1 1 | 27 | 729 |
| 4 | 1 0 0 0 0 | 1 0 1 0 0 | 18 | 324 |
| Sum | | | | 2354 |
| Average | | | | 588.5 |
| Max | | | | 729 |

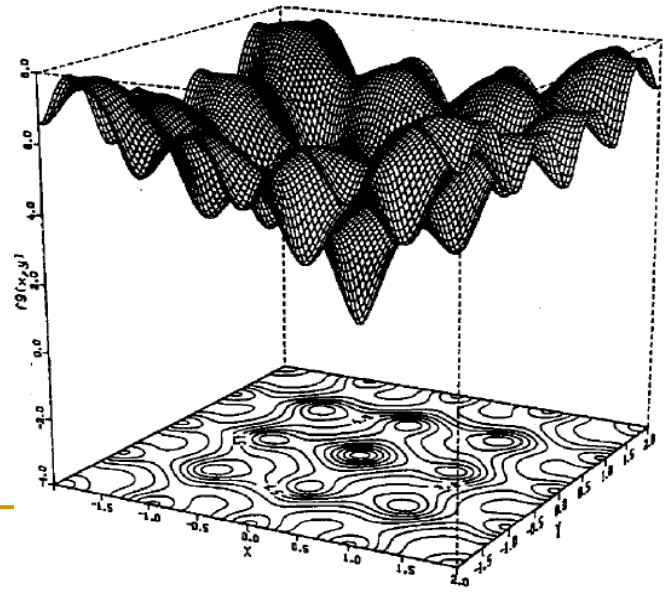
# Hard Problems

- Solution of $f(x)=x^2$ is easy to find and not realistic for GA
- Many problems occur as real valued problems, e.g. continuous parameter optimization $f : \mathcal{R}^n \rightarrow \mathcal{R}$
- Illustration: Ackley's function (often used in EC)

$$f(\overline{x}) = -c_1 \cdot exp\left(-c_2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$$

$$-exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} cos(c_3 \cdot x_i)\right) + c_1 + 1$$

$$c_1 = 20,\ c_2 = 0.2,\ c_3 = 2\pi$$



# Genetic Algorithms To Solve The Traveling Salesman Problem (TSP)

# The Problem

The **Traveling Salesman Problem** is defined as:

*'We are given a set of cities and a symmetric distance matrix that indicates the cost of travel from each city to every other city.*

*The goal is to find **the shortest circular tour,** visiting every city exactly once, so as to **minimize the total travel cost**, which includes the cost of traveling from the last city back to the first city'.*

# Encoding

- I represent every city with an integer .

- Consider 6 Indian cities –

  Mumbai, Nagpur , Calcutta, Delhi , Bangalore and Chennai and assign a number to each.

Mumbai → 1
Nagpur → 2
Calcutta → 3
Delhi → 4
Bangalore → 5
Chennai → 6

# Encoding (contd.)

- Thus a path would be represented as a sequence of integers from 1 to 6.
- The path [1 2 3 4 5 6 ] represents a path from Mumbai to Nagpur, Nagpur to Calcutta, Calcutta to Delhi, Delhi to Bangalore, Bangalore to Chennai, and finally from Chennai to Mumbai.
- This is an example of **Permutation Encoding** as the position of the elements determines the fitness of the solution.

# Fitness Function

- The fitness function will be the total cost of the tour represented by each chromosome.

- This can be calculated as the sum of the distances traversed in each travel segment.

*The Lesser The Sum, The Fitter The Solution Represented By That Chromosome.*

# Distance/Cost Matrix For TSP

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 863 | 1987 | 1407 | 998 | 1369 |
| 2 | 863 | 0 | 1124 | 1012 | 1049 | 1083 |
| 3 | 1987 | 1124 | 0 | 1461 | 1881 | 1676 |
| 4 | 1407 | 1012 | 1461 | 0 | 2061 | 2095 |
| 5 | 998 | 1049 | 1881 | 2061 | 0 | 331 |
| 6 | 1369 | 1083 | 1676 | 2095 | 331 | 0 |

**Cost matrix for six city example.**

*Distances in Kilometers*

# Fitness Function (contd.)

- So, for a chromosome [4 1 3 2 5 6], the total cost of travel or fitness will be calculated as shown below

- Fitness = *1407 + 1987 + 1124 + 1049 + 331+ 2095*
  = *7993 kms.*

- Since our objective is to Minimize the distance, the lesser the total distance, the fitter the solution.

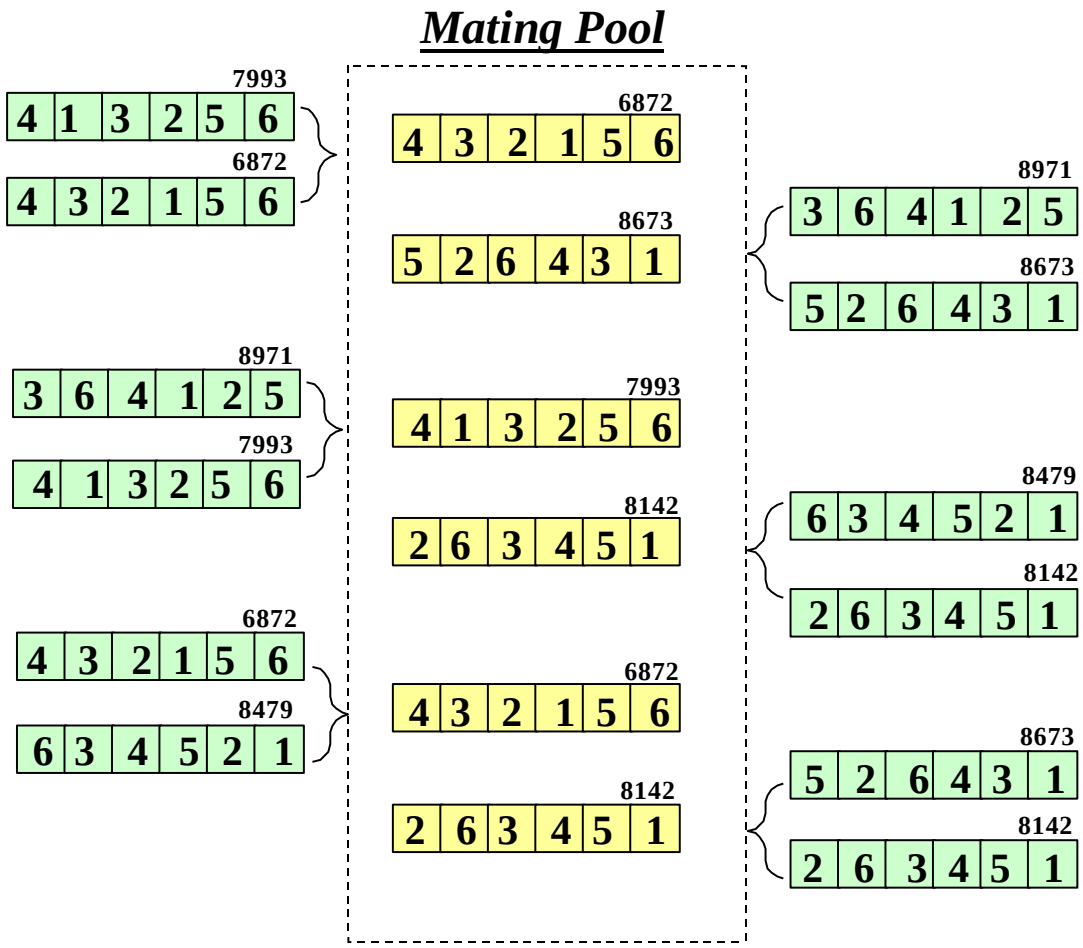
# Selection Operator

May use ***Tournament Selection***.

As the name suggests *tournaments* are played between two solutions and the better solution is chosen and placed in the *mating pool*.

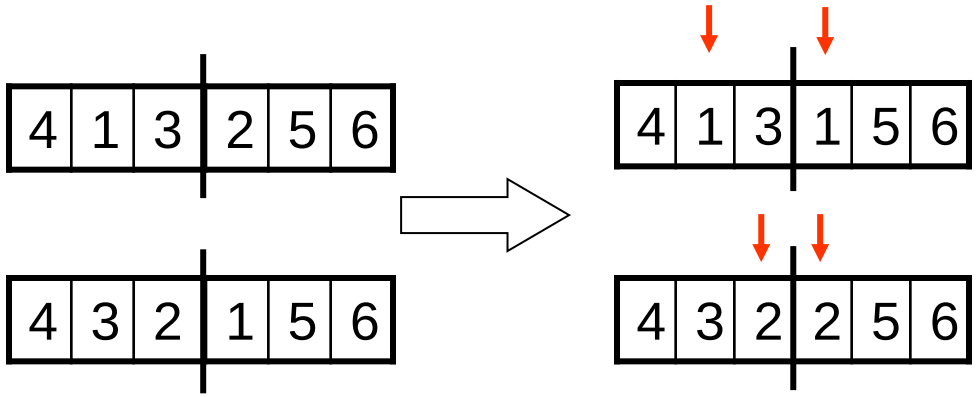
Two other solutions are picked again and another slot in the *mating pool* is filled up with the better solution.

# Tournament Selection (contd.)



# Why we cannot use single-point crossover:

- Single point crossover method randomly selects a crossover point in the string and swaps the substrings.
- This may produce some invalid offsprings as shown below.



# Crossover Operator

- Used the *Enhanced Edge Recombination* operator (T.Starkweather, et al, *'A Comparison of Genetic Sequencing Operators,* International Conference of GAs, 1991).

- This operator is different from other genetic sequencing operators in that it emphasizes *adjacency information* instead of the order or position of items in the sequence.

- The algorithm for the Edge-Recombination Operator involves constructing an Edge Table first.

# Edge Table

The *Edge Table* is an *adjacency table* that lists links *into* and *out of* a city found in the two parent sequences.

If an item is already in the edge table and we are trying to insert it again, that element of a sequence must be a *common edge* and is represented by inverting it's sign.

# Finding The Edge Table

| | | | | | |
|---|---|---|---|---|---|
| Parent 1 | 4 | 1 | 3 | 2 | 5 | 6 |
| Parent 2 | 4 | 3 | 2 | 1 | 5 | 6 |

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 3 | 2 | 5 |
| 2 | -3 | 5 | 1 | |
| 3 | 1 | -2 | 4 | |
| 4 | -6 | 1 | 3 | |
| 5 | 1 | 2 | -6 | |
| 6 | -5 | -4 | | |

# Enhanced Edge Recombination Algorithm

1. Choose the initial city from one of the two parent tours. (It can be chosen randomly as according to criteria outlined in *step 4*). This is the *current city.*

2. Remove all occurrences of the *current city* from the left hand side of the edge table.( These can be found by referring to the edge-list for the *current city*).

3. If the *current city* has entries in it's edge-list, go to s*tep 4* otherwise go to *step 5*.

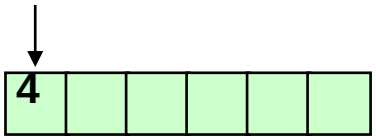
4. Determine which of the cities in the edge-list of the *current city* has the fewest entries in it's own edge-list. The city with fewest entries becomes the *current city*. In case a negative integer is present, it is given preference. Ties are broken randomly. Go to *step 2.*

5. If there are no remaining *unvisited* cities, then *stop.* Otherwise, randomly choose an *unvisited* city and go to *step 2.*

# Example Of Enhanced Edge Recombination Operator

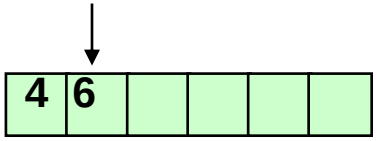
**Step 1**

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 3 | 2 | 5 |
| 2 | -3 | 5 | 1 | |
| 3 | 1 | -2 | 4 | |
| 4 | -6 | 1 | 3 | |
| 5 | 1 | 2 | -6 | |
| 6 | -5 | -4 | | |



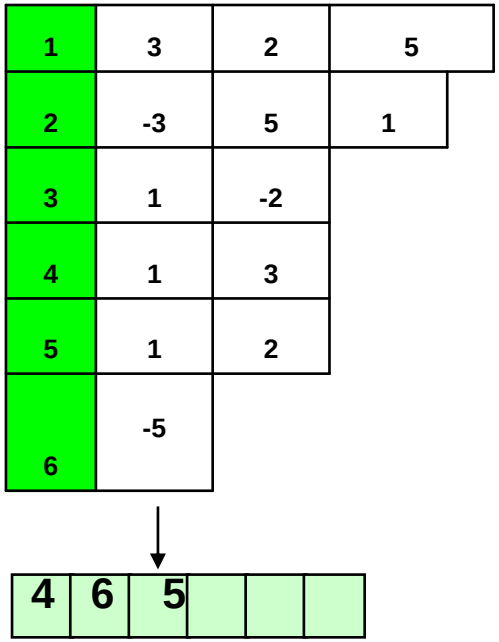

**Step 2**

| | | | |
|---|---|---|---|
| 1 | 3 | 2 | 5 |
| 2 | -3 | 5 | 1 |
| 3 | 1 | -2 | |
| 4 | -6 | 1 | 3 |
| 5 | 1 | 2 | -6 |
| 6 | -5 | | |

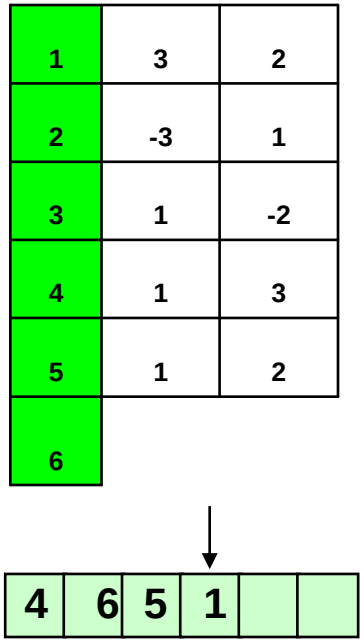


# Example Of Enhanced Edge Recombination Operator (contd.)

**Step 3**

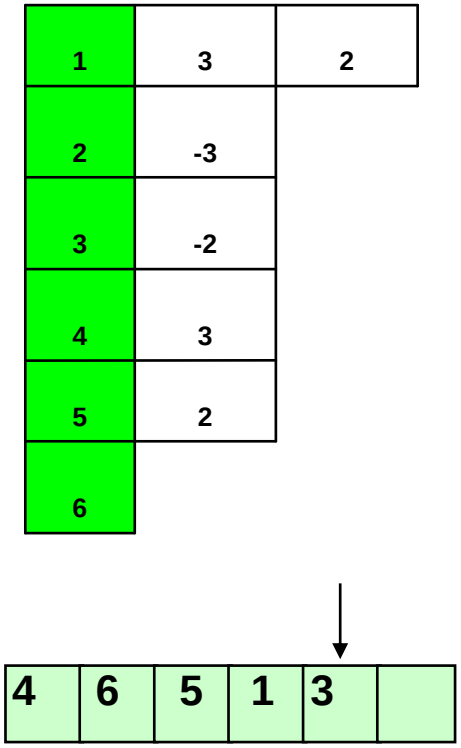



**Step 4**

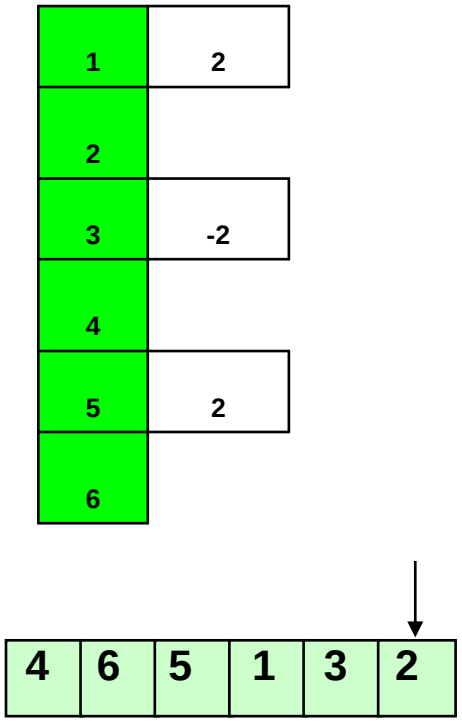


# Example Of Enhanced Edge Recombination Operator (contd.)

**Step 5**

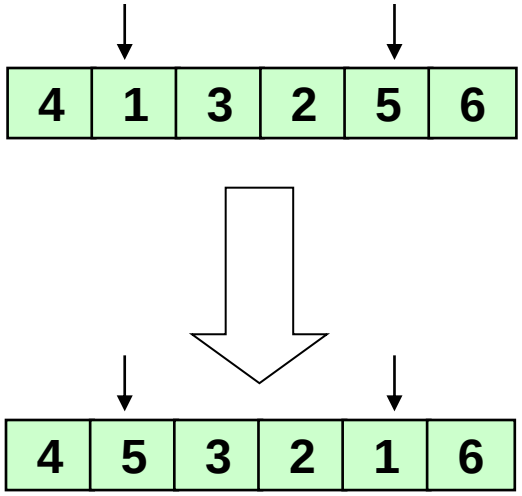


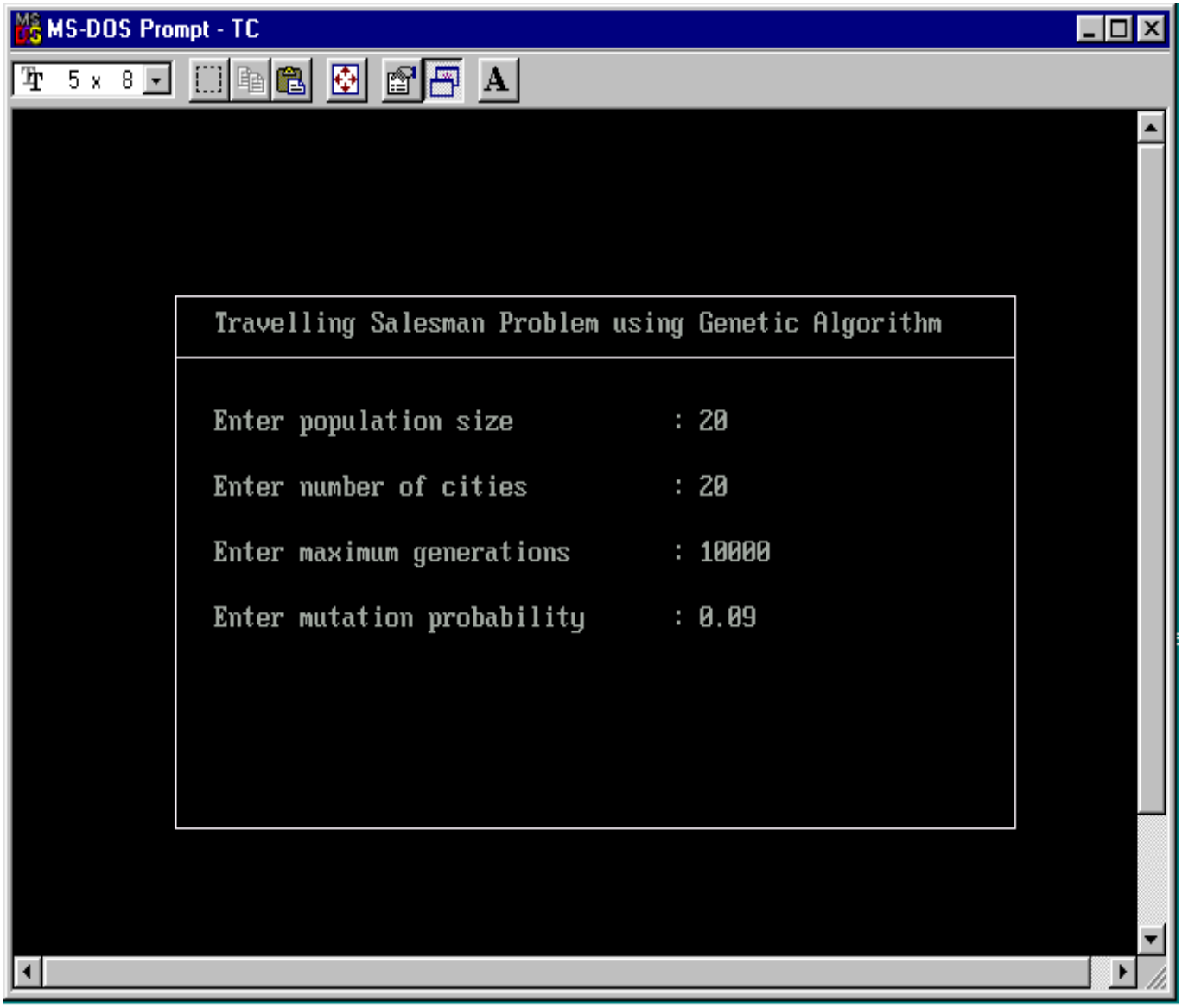

**Step 6**



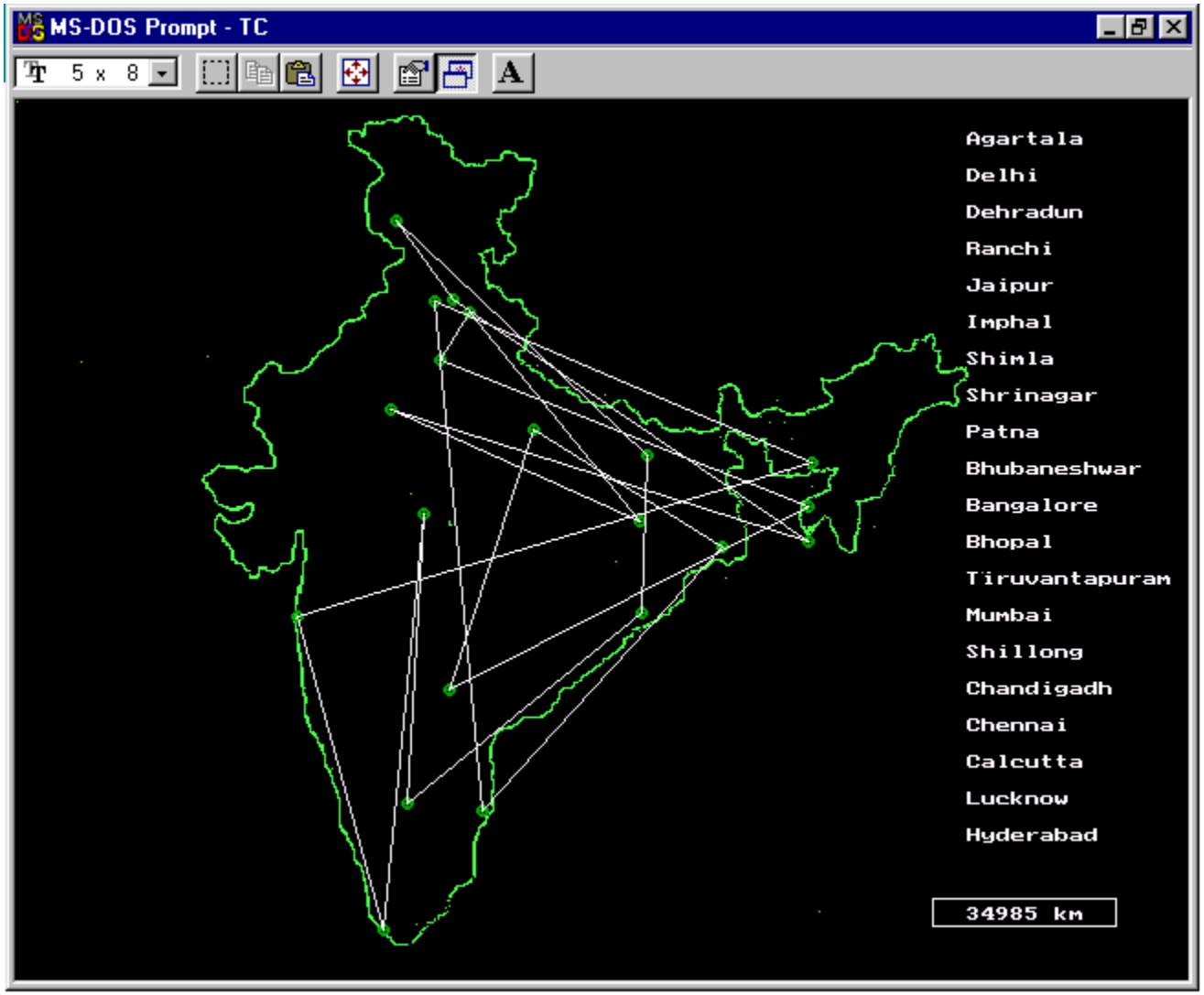
# Mutation Operator

- The mutation operator induces a change in the solution, so as to maintain diversity in the population and prevent *Premature Convergence*.
- In our project, we mutate the string by randomly selecting any two cities and interchanging their positions in the solution, thus giving rise to a new tour.

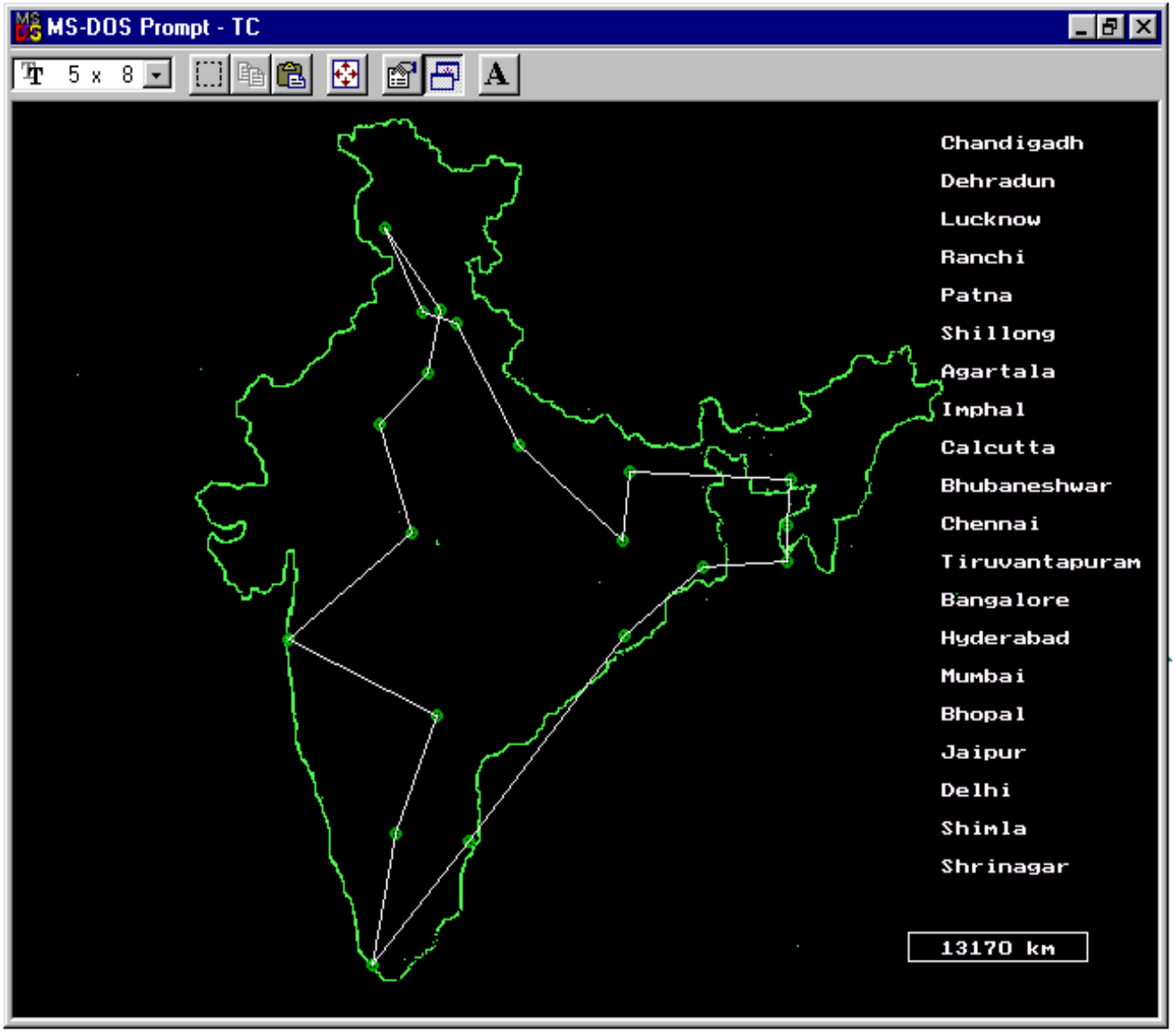





# Input To Program




**Initial Output For 20 cities : Distance=34985 km**
**Initial Population**



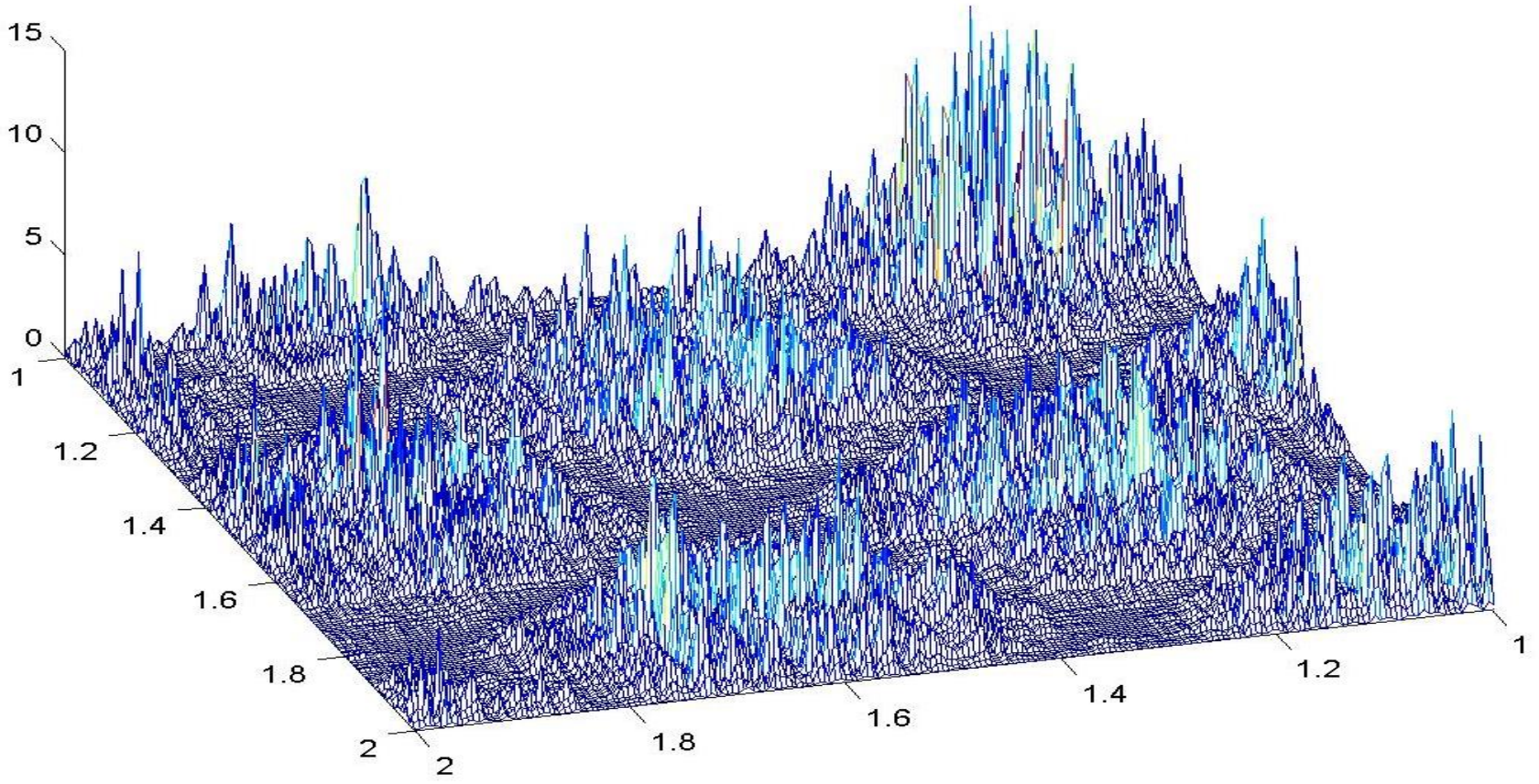

**Final Output For 20 cities : Distance=13170 km**
**Generation 4786**

# Other Typical applications:

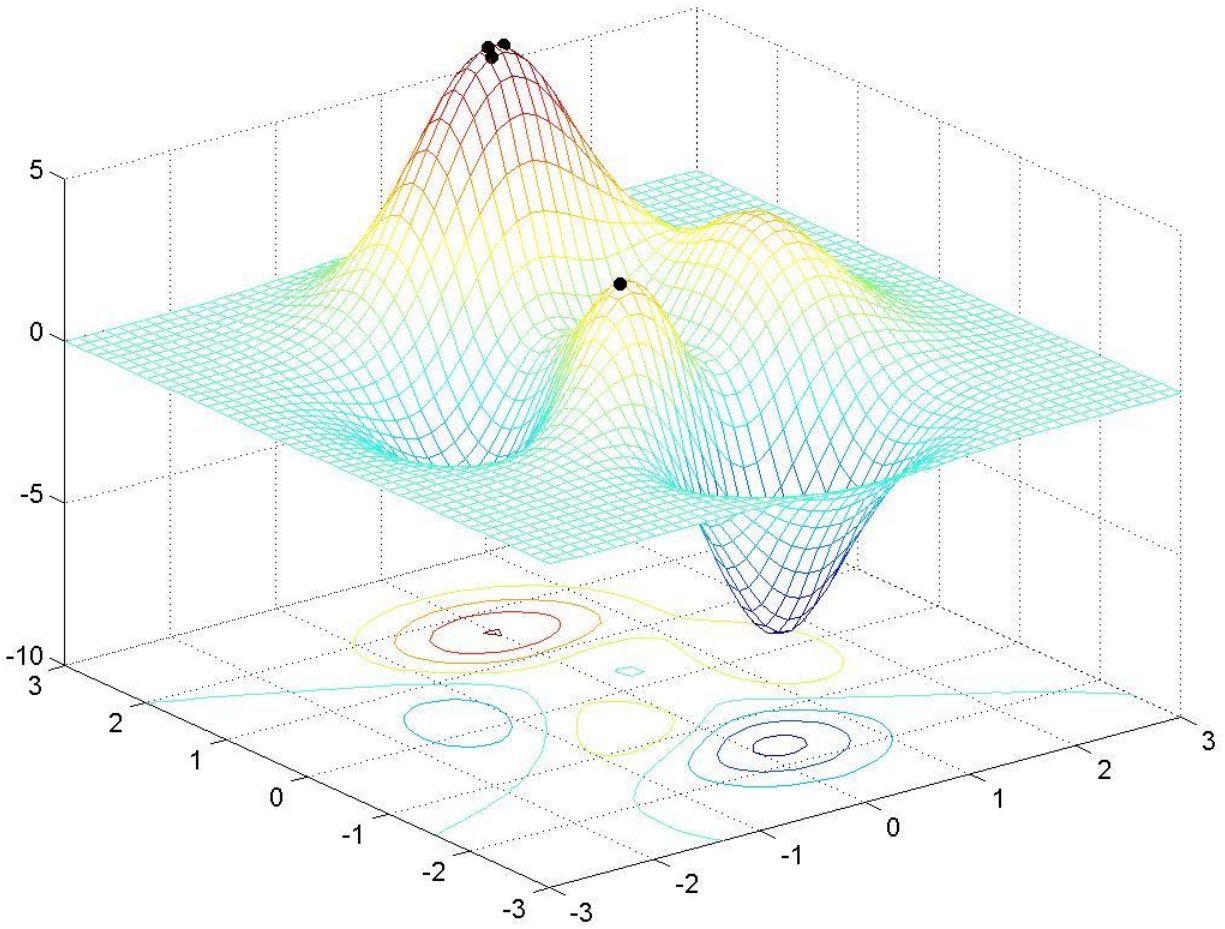
**When to use it?**

- machine learning
- timetabling, scheduling
- data mining
- robot trajectory
- etc.
- optimisation problems
- designing neural networks
- strategy planning
- evolving programs

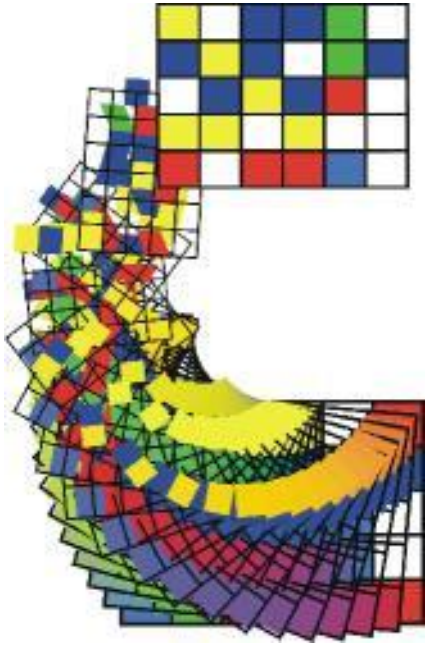
# *Extremes of multimodal functions*



# *Extremes of multimodal functions*



# Timetabling



- Each chromosome represents a timetable
- A random population of feasible
- timetables is created
- Each timetable is evaluated according to chosen criteria
- Timetables are selected for reproduction
- Crossover and mutation operators are used to produce offspring
- The population increasingly consists of “good” timetables

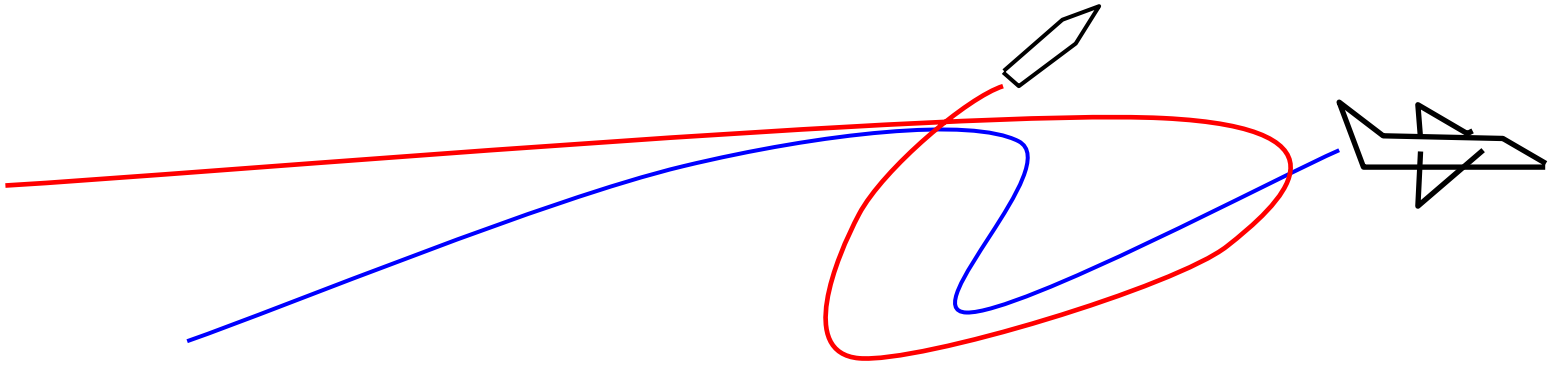
# Evolution of Strategies - Game Playing

- Each chromosome represents different strategies for playing the game
- Initial population is generated randomly
- During the selection process each strategy is required to play a certain number of games against other strategies
- The strategies with the most wins are selected for reproduction
- The population increasingly consists of “good” strategies

# Rule Discovery

- SAMUEL system discovers rules by which a slower but more manoeuvrable aircraft can evade a faster but less agile missile until the missile runs out of fuel
- The aircraft can sense the range, speed, heading and bearing of the missile
- Rules are of a fairly uniform sort, such as to turn left by 90 degrees if the missile parameters are lie within certain intervals

# Rule Discovery

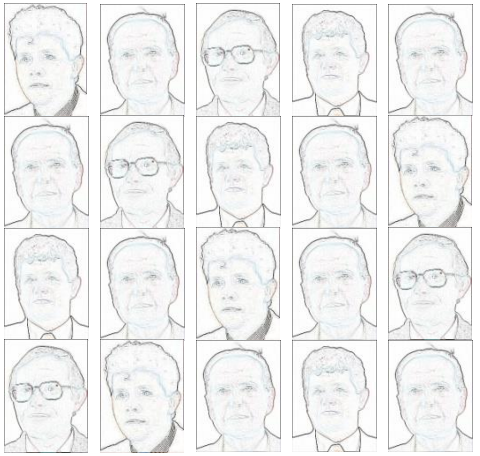


- Chromosomes are ordered sets of possible rules (if situation x occurs do action encoded within x-th gene)
- Fitness evaluation is by simulation (it is examined for how long the aircraft can evade the missile using particular set of rules)
- It takes hours to evaluate initial population !

# Criminal suspect recognition

- It is easy to identify a criminal from photo but hard to describe his or her features
- It is difficult to generate even from computer library of visual features
- Idea to use genetic algorithms to help witnesses in the identification of criminal suspects

# Criminal suspect recognition



***Faceprints***

- randomly generates 20 faces on a computer screen
- witness evaluates each face on a 10 point scale
- GA generates additional faces from 5 building blocks: eyes, mouth, nose, hair and chin (7-bit string each)
- Chromosome is a 35-bit binary string (34 billion faces)
- Witness rates successive generations with 10 point scale
- Convergence often occurs after 20 generations

# Boeing 777

- The engines were designed by classical techniques with the emphasis on efficient fuel consumption
- Genetic algorithms were used to fine-tuning of some parameters of the already designed engines
- Due to their utilization at this stage the consumption of fuel has been reduced by 2,5%
- (operating cost savings cca 2 mil USD at one plane per year)

# Summary

- *Genetic Algorithms* (GAs) implement optimization strategies based on simulation of the natural law of evolution of a species by *natural selection*

- The basic GA Operators are:
  - Encoding
  - Recombination
  - Crossover
  - Mutation

- GAs have been applied to a variety of function optimization problems, and have been shown to be highly effective in searching a large, poorly defined search space even in the presence of difficulties such as high-dimensionality, multi-modality, discontinuity and noise.

# Questions ?

# Particle Swarm Optimization (PSO)


Dr. Md. Aminul Haque Akhand, Dept. of CSE, KUET

# Introduction : Swarm Intelligence (SI)



- Study of collective behavior in decentralized, self-organized systems.
- Originated from the study of colonies, or swarms of social organisms.
- Collective intelligence arises from interactions.

# Overview of Particle Swarm Optimization(PSO)

PSO is a population based on stochastic optimization algorithms to find a solution and then solve an optimization problem in a search space.

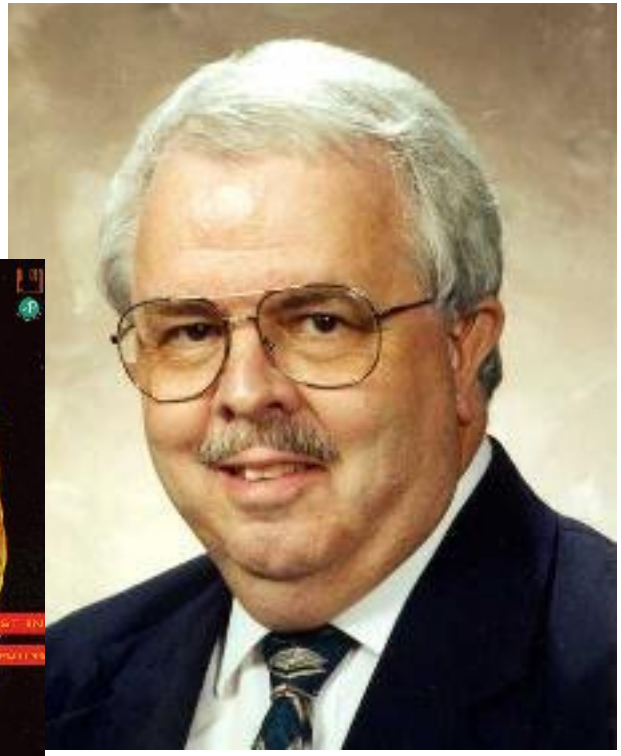
How can birds or fish exhibit such a coordinated collective behavior?

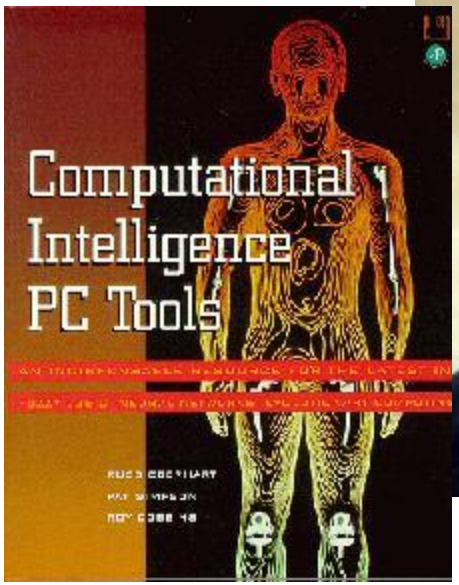
# PSO: History and Properties



- Particle Swarm Optimization:
  - Introduced by Kennedy & Eberhart 1995
  - Inspired by social behavior of birds and shoals of fish
  - Swarm Intelligence-based optimization
  - Population-based optimization
  - Nondeterministic
  - Performance comparable to Genetic Algorithms

# PSO Inventors

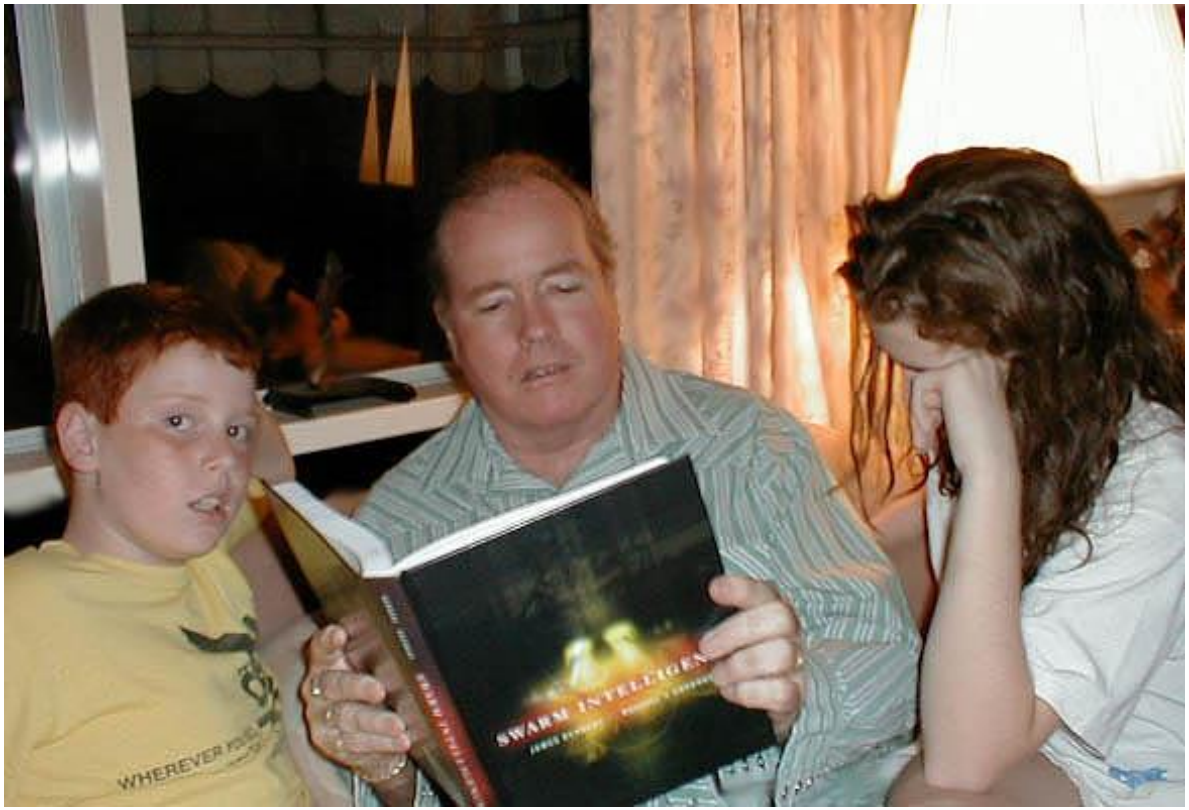





Russell Eberhart

eberhart@engr.iupui.edu

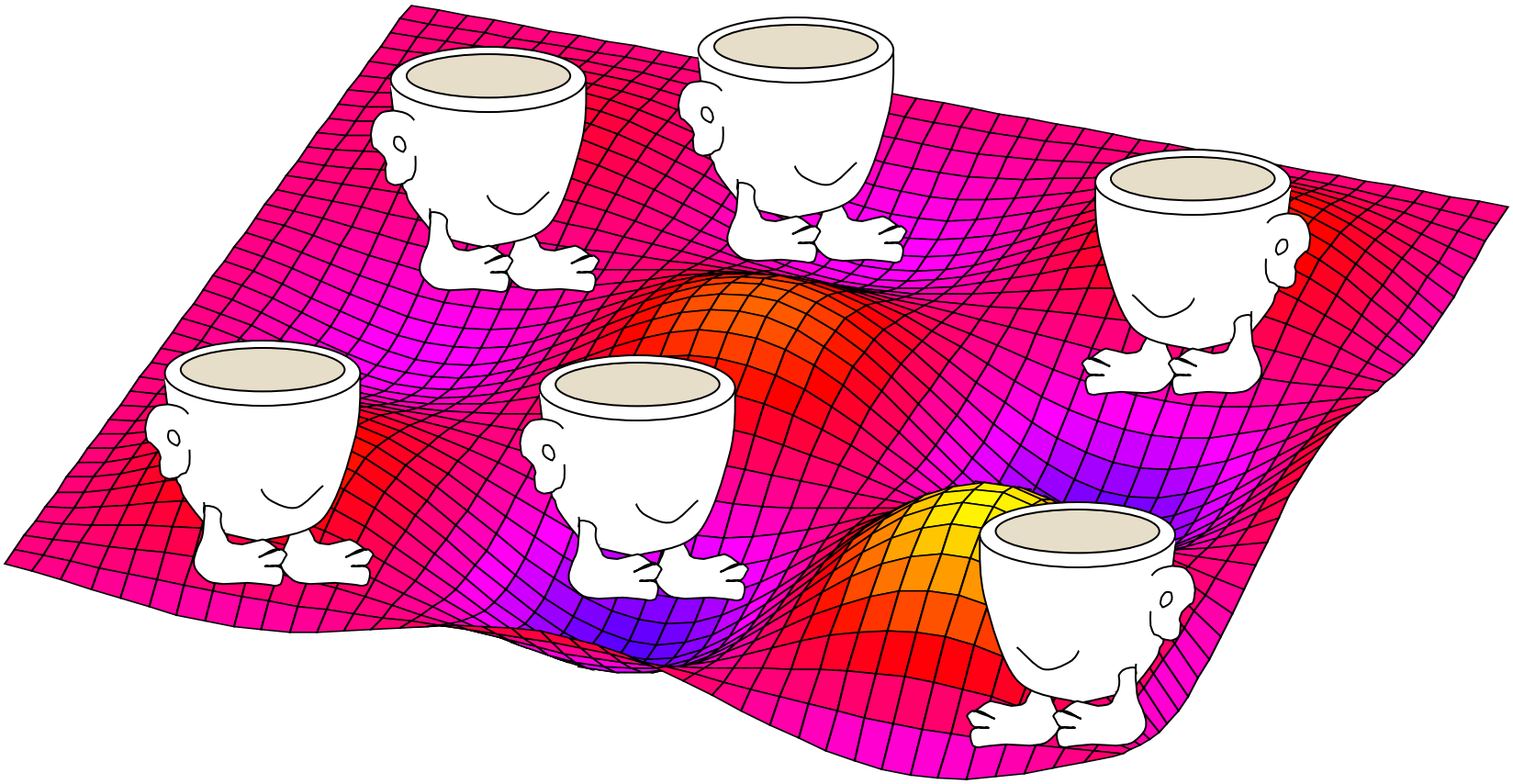
# PSO Inventors (2)




James Kennedy

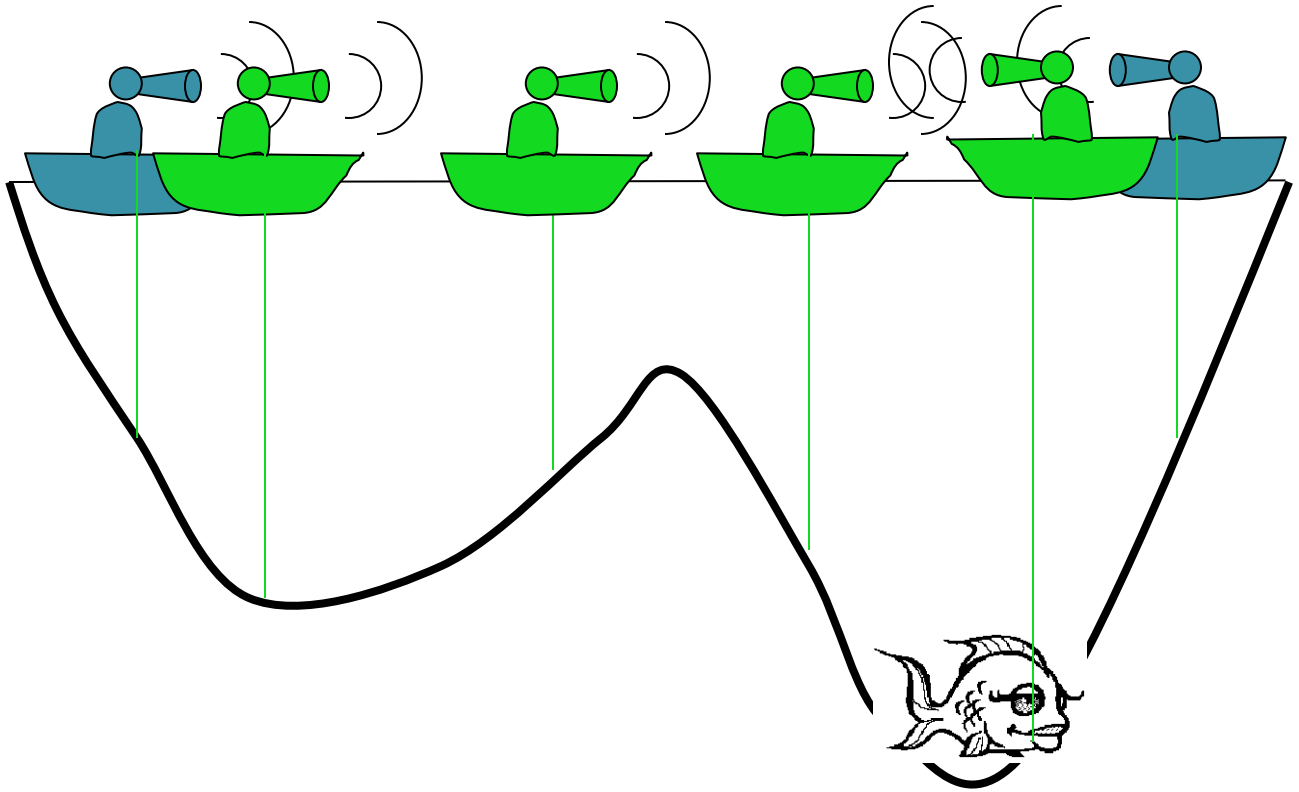
Kennedy_Jim@bls.gov

# PSO: United We Stand



Maurice.Clerc@WriteMe.com 2002-04-24

# Cooperation Example for PSO



# PSO Formulation



- Swarm : a set of particles (S)
- Particle: a potential solution
  - Position, $X_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in \mathbb{R}^n$
  - Velocity , $V_i = (v_{i1}, v_{i2}, \ldots, v_{in}) \in \mathbb{R}^n$
- Each particle maintains
  - Individual best position: $P_i = (p_{i1}, p_{i2}, \ldots, p_{in}) \in \mathbb{R}^n$, $pbest_i = f(P_i)$
- Swarm maintains its global best: $P_g \in \mathbb{R}^n$, $gbest = f(P_g)$

# PSO Algorithm



- Basic Algorithm of PSO:
  1. Initialize the swarm from the solution space
  2. Evaluate fitness of each particle
  3. Update individual and global bests
  4. Update velocity and position of each particle
  5. Go to Step 2, and repeat until termination condition

# PSO Algorithm (cont.)



- **Original velocity update equation:**

$$V_i^{t+1} = \underbrace{V_i^t}_{\text{Inertia}} + \underbrace{\varphi_1 . r_1 (P_i - X_i^t)}_{\text{Cognitive Component}} + \underbrace{\varphi_2 . r_2 (P_g - X_i^t)}_{\text{Social Component}}$$

  - with $r_1, r_2 \sim U(0,1)$
  - $\varphi_1, \varphi_2$ : acceleration constant

# PSO Algorithm (cont.)



- **Original velocity update equation:**

$$V_i^{t+1} = V_i^t + \varphi_1 . r_1 (P_i - X_i^t) + \varphi_2 . r_2 (P_g - X_i^t)$$

Inertia | Cognitive Component | Social Component

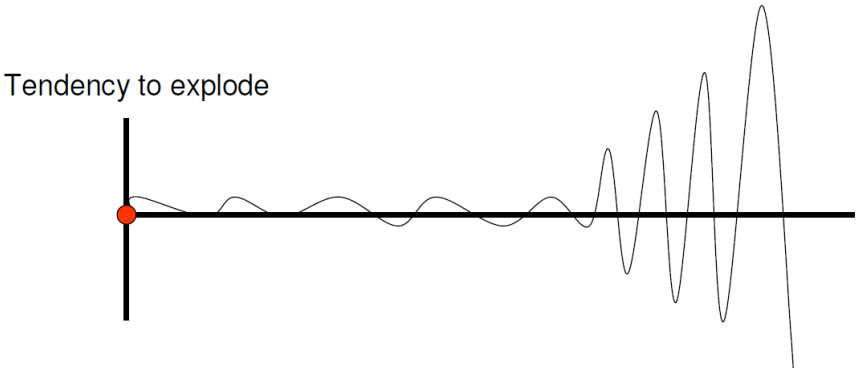
  - with $r_1, r_2 \sim U(0,1)$
  - $\varphi_1, \varphi_2$ : Acceleration constants or coefficients
  - Acceleration constants sometimes define as $C_1$ and $C_2$

- **Position Updat** $X_i^{t+1} = X_i^t + V_i^{t+1}$

# PSO Algorithm - Parameters



- Acceleration constant $\varphi_1, \varphi_2$
  - Small values limit the movement of the particles
  - Large values : tendency to explode toward infinity
  - In general $\varphi_1 + \varphi_2 \leq 4$




- Maximum velocity
  - Velocity is a stochastic variable => uncontrolled trajectory

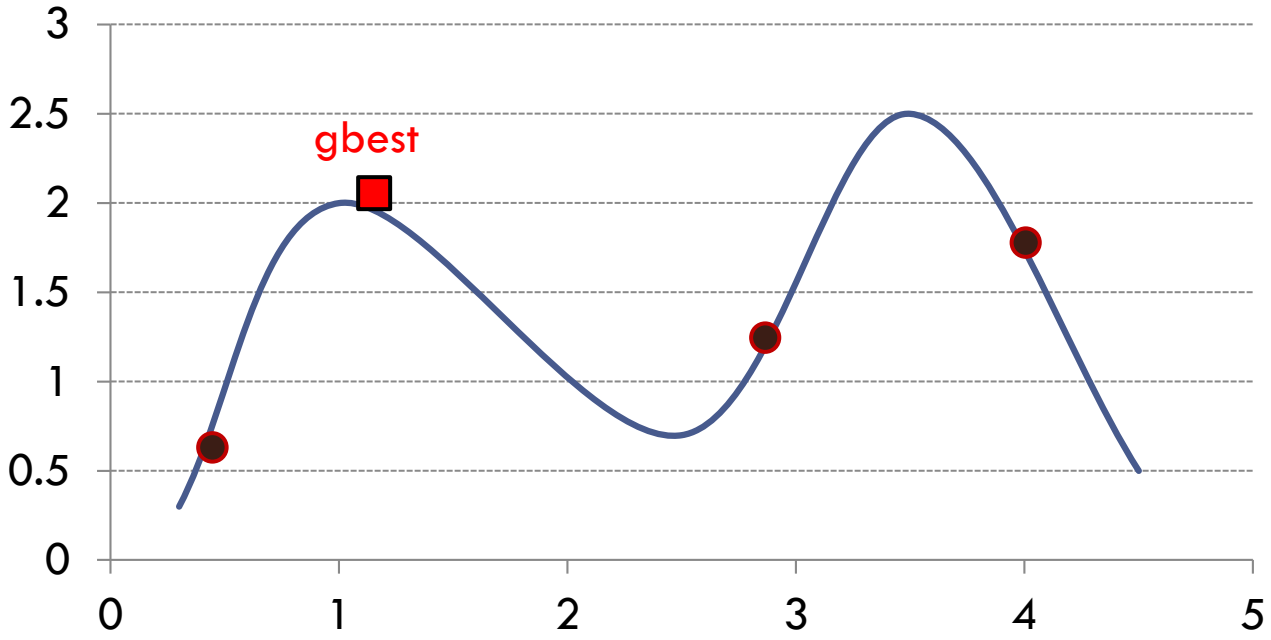
$$\text{If} \quad v_{\text{id}} > v_{\text{max}} \text{ then } v_{\text{id}} = v_{\text{max}}$$

$$\text{else if } v_{\text{id}} < -v_{\text{max}} \text{ then } v_{\text{id}} = -v_{\text{max}}$$

# Simple 1D Example



Initialize swarm and evaluate fitness (t=0)

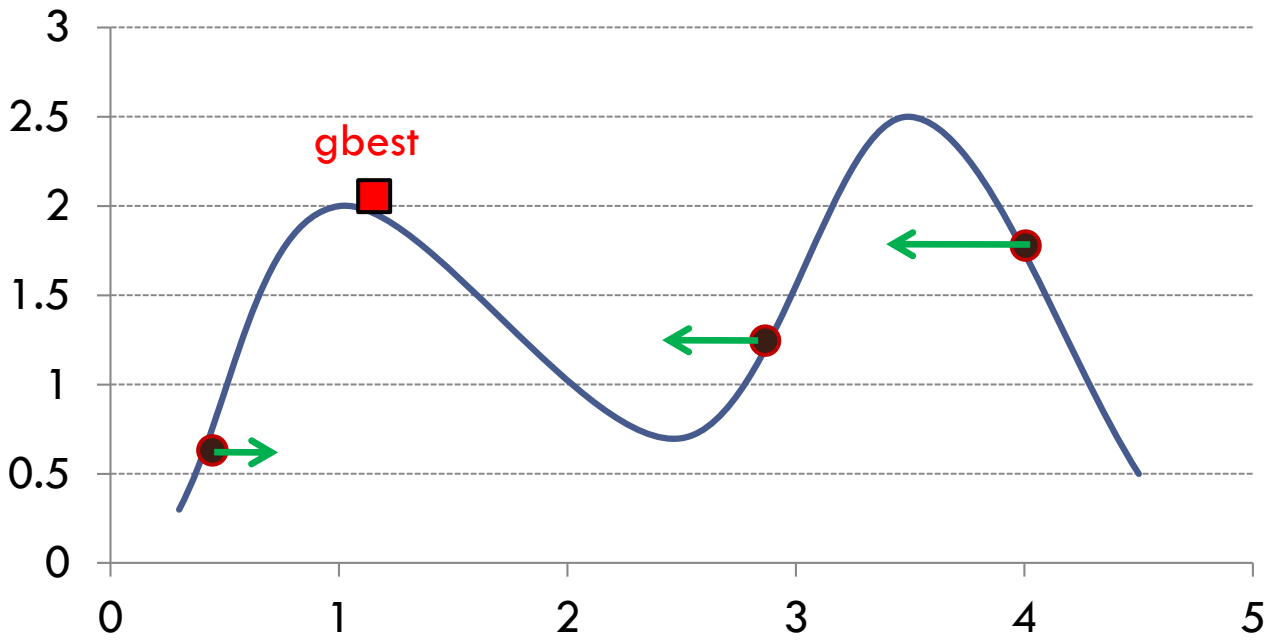


# Simple 1D Example



Update velocity and position (t=1)

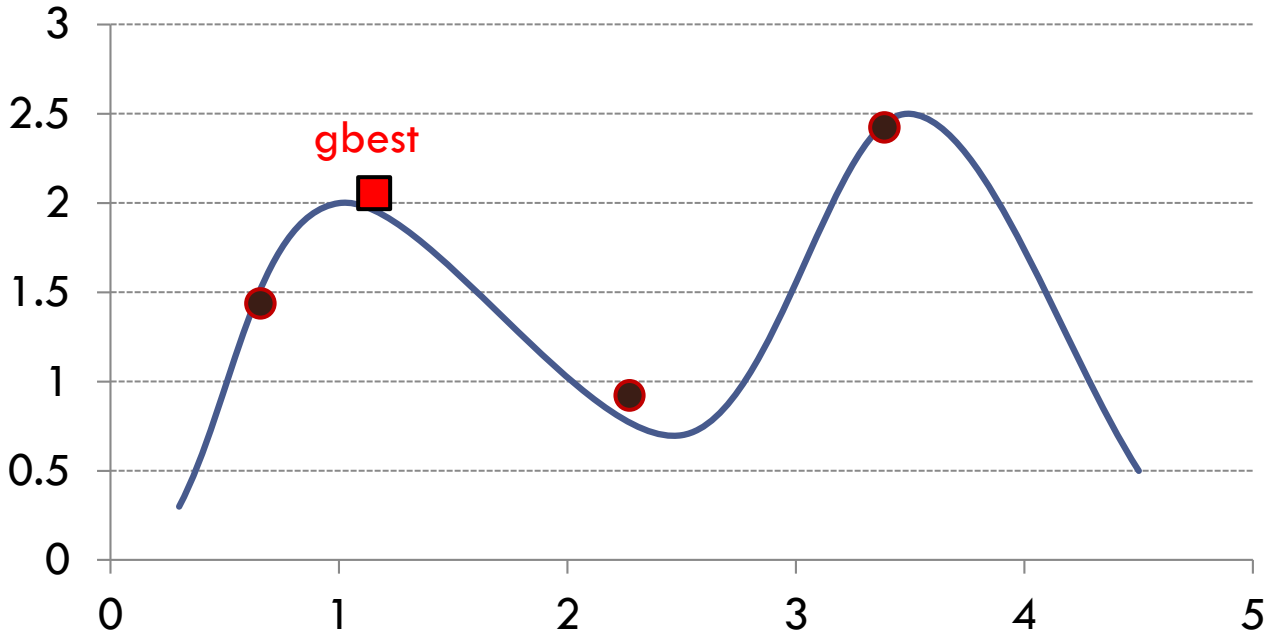
$$V_i^{t+1} = V_i^t + \varphi_1 . r_1 (P_i - X_i^t) + \varphi_2 . r_2 (P_g - X_i^t)$$



# Simple 1D Example



Evaluate fitness
Update personal and global best (t=2)

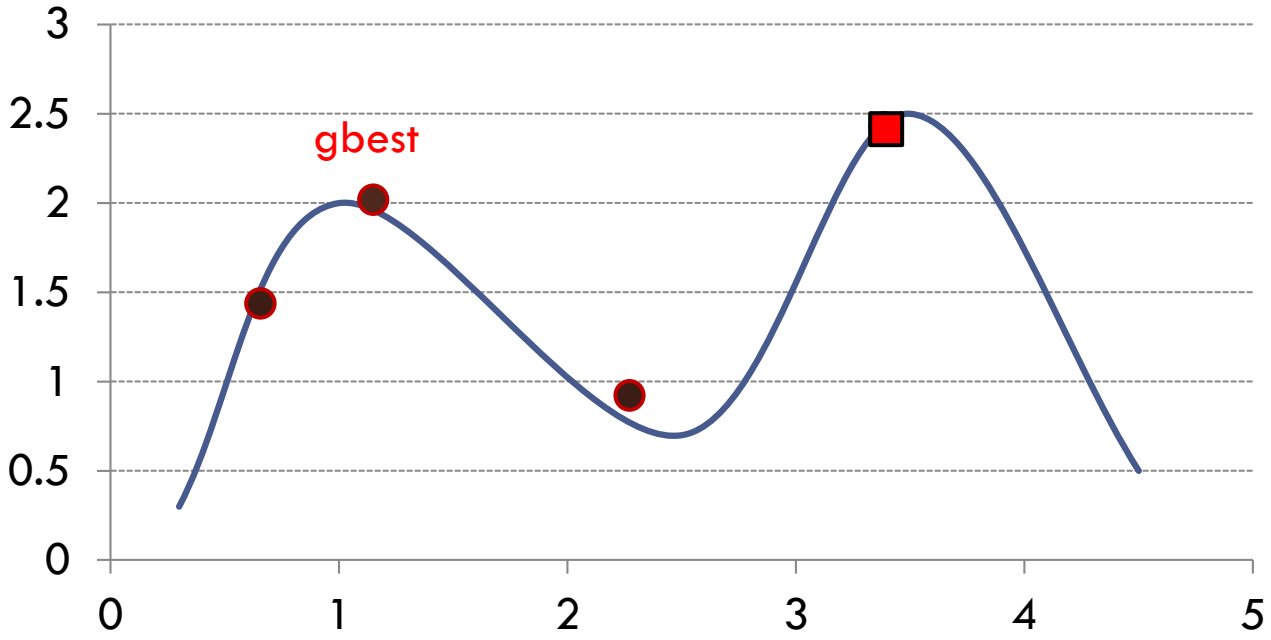


# Simple 1D Example



Evaluate fitness

Update personal and global best (t=2)

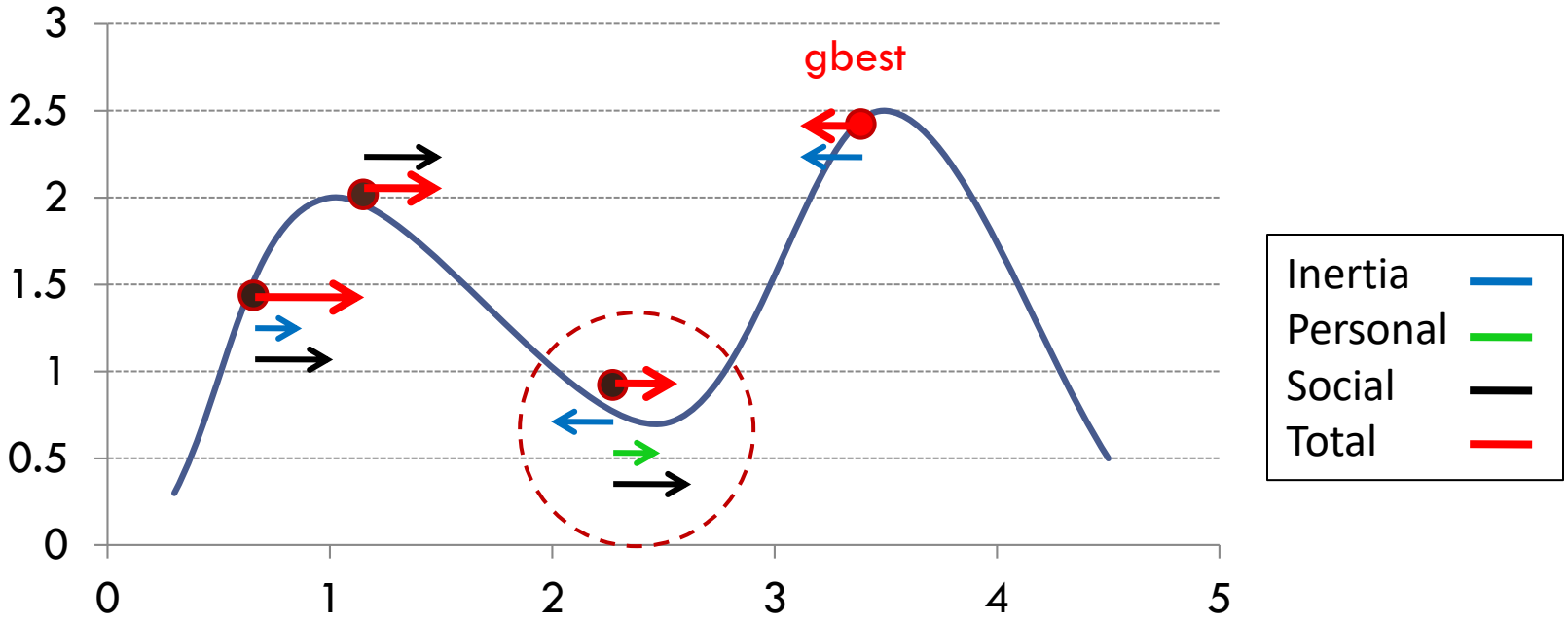


# Simple 1D Example



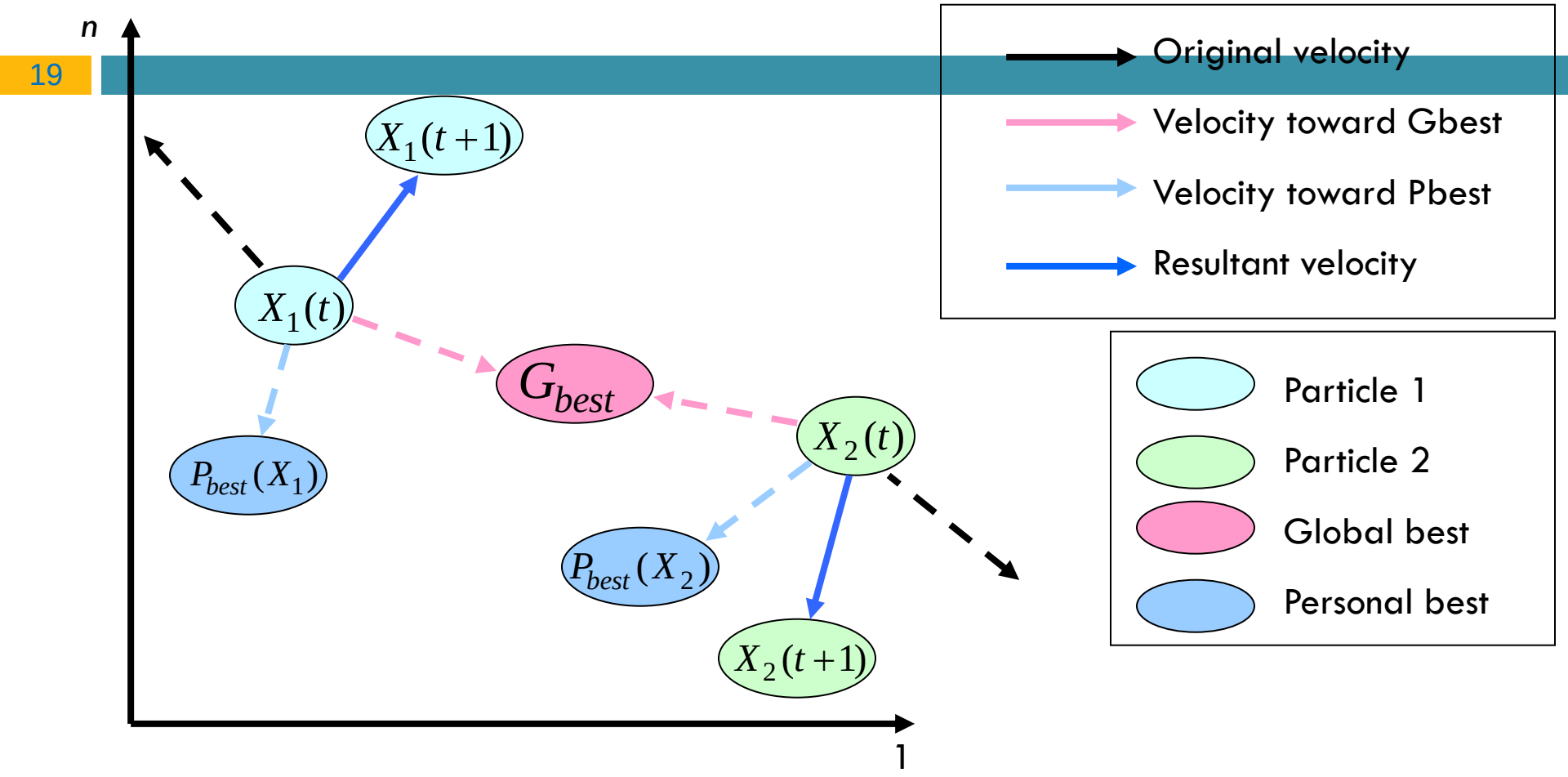
Update velocity and position (t=2)

$$V_i^{t+1} = V_i^t + \varphi_1 . r_1 (P_i - X_i^t) + \varphi_2 . r_2 (P_g - X_i^t)$$



# Movement of Particles






Individual particles (1 and 2) are accelerated toward the location of the global best solution (Gbest) and the location of their own personal best (Pbest) in the *n*-dimensional space.

# PSO with Inertia Weight (w)



- **Inertia weight:**

$$V_i^{t+1} = w.V_i^t + \varphi_1.r_1(P_i - X_i^t) + \varphi_2.r_2(P_g - X_i^t)$$

- Scaling the previous velocity

- **Rate of Convergence Improvement**

- Control search behavior
  - High values → exploration
  - Low values → exploitation

# PSO with Inertia Weight (w) (Cont.)



- Can be decreased over time:
  - Linear [0.9 to 0.4]
  - Nonlinear

$$w(t) = \frac{A}{e^t}$$

- Main disadvantage:
  - Once the inertia weight is decreased, the swarm loses its ability to search new areas (can not recover its exploration mode).

# Rate of Convergence Improvement (Cont.)



- **Constriction Factor:**
  - Canonical PSO

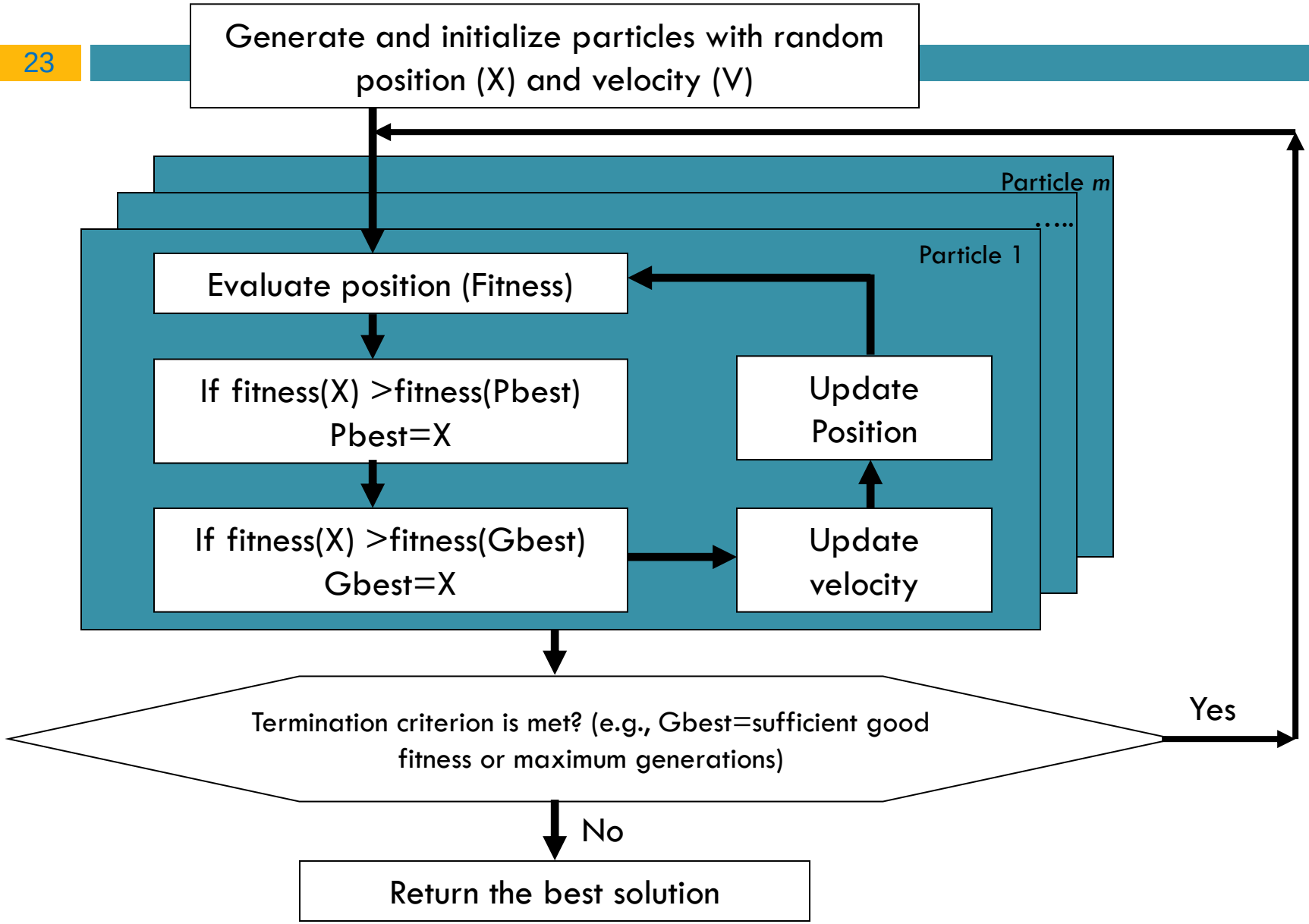
$$V_i^{t+1} = \chi.\left(V_i^t + \varphi_1.r_1(P_i - X_i^t) + \varphi_2.r_2(P_g - X_i^t)\right)$$

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad , \quad \varphi = \varphi_1 + \varphi_2 \, , \varphi > 4$$

  - Typically $\varphi = 4.1 \quad \chi = 0.729$
  - Can converge without using Vmax (velocity clamping)
  - Improve the convergence by damping the oscillations

# The Flowchart of PSO





# Overview of PSO with Pseudocode

```
For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest) in history
            set current value as the new pBest
    End

    Choose the particle with the best fitness value of all the particles as the gBest
    For each particle
        Calculate particle velocity according equation of updating velocity
        Update particle position according equation of updating position
    End
While maximum iterations or minimum error criteria is not attained
```

Pseudo-code for PSO

# Aspects of PSO

$$V_i^{t+1} = w.V_i^t + \varphi_1.r_1(P_i - X_i^t) + \varphi_2.r_2(P_g - X_i^t)$$



- **Previous Velocity, $V_i^t$**
  - ✓ Inertia component
  - ✓ Memory of previous flight direction
  - ✓ Prevents particle from drastically changing direction
- **Cognitive Component, $\varphi_1.r_1(P_i - X_i^t)$**
  - ✓ Quantifies performance relative to past performances
  - ✓ Memory of previous best position
  - ✓ Nostalgia
- **Social Component, $\varphi_2.r_2(P_g - X_i^t)$**
  - ✓ Quantifies performance relative to neighbors
  - ✓ Envy

# Aspects of PSO (Cont.)

$$V_i^{t+1} = w.V_i^t + \varphi_1.r_1(P_i - X_i^t) + \varphi_2.r_2(P_g - X_i^t)$$



**Inertia weight (w) Controls the tendency of particles to keep searching in the same direction**

- **For w ≥ 1**
  - ✓ Velocities increase over time
  - ✓ Swarm diverges
  - ✓ Particles fail to change direction towards more promising regions

- **For 0 < w < 1**
  - ✓ Particles decelerate
  - ✓ Convergence also depend on values of $\varphi_1$ and $\varphi_2$

- **Exploration–Exploitation**
  - Large values – Favor exploration
  - Small values – Promote exploitation

- **Problem-dependent**

# Aspects of PSO (Cont.)

$$V_i^{t+1} = w.V_i^t + \varphi_1.r_1(P_i - X_i^t) + \varphi_2.r_2(P_g - X_i^t)$$



Exploration–exploitation tradeoff

- exploration – the ability to explore regions of the search space
- exploitation – the ability to concentrate the search around a promising area to refine a candidate solution
- $c_1$ vs $c_2$ ($\varphi_1$ vs $\varphi_2$) influence on the exploration–exploitation tradeoff

# Aspects of PSO (Cont.)

$$V_i^{t+1} = w.V_i^t + \varphi_1.r_1(P_i - X_i^t) + \varphi_2.r_2(P_g - X_i^t)$$



- $\varphi_1$ and $\varphi_2$, together with $r_1$ and $r_2$, control the stochastic influence of the cognitive and social components on the overall velocity of a particle.
- The acceleration constants $\varphi_1$ and $\varphi_2$ are also referred to as trust parameters, where $\varphi_1$ expresses how much confidence a particle has in itself, while $\varphi_2$ expresses how much confidence a particle has in its neighbors.
- If $\varphi_1 > 0$ and $\varphi_2 = 0$, all particles are independent hill-climbers. Each particle finds the best position in its neighborhood by replacing the current best position if the new position is better. Particles perform a local search.
- If $\varphi_2 > 0$ and $\varphi_1 = 0$, the entire swarm is attracted to a single point $P_g$

# Aspects of PSO (Comparing with GA)



- PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA).
- The system is initialized with a population of random solutions and searches for optima by updating generations.
- However, unlike GA, PSO has no evolution operators such as crossover and mutation.
- In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.
- Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust.
- PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

# PSO Variants



- Hybrid PSO
  - Incorporate the capabilities of other evolutionary computation techniques.
- Adaptive PSO
  - Adaptation of PSO parameters for a better performance.
- PSO in complex environments
  - Multiobjective or constrained optimization problems or tracking dynamic systems.
- Other variants
  - variations to the original formulation to improve its performance.

# Hybrid PSO



- GA-PSO:
  - combines the advantages of swarm intelligence and a natural selection mechanism.
  - jump from one area to another by the selection mechanism → accelerating the convergence speed.
  - capability of “breeding”.
  - replacing agent positions with low fitness values, with those with high fitness, according to a selection rate

# Hybrid PSO



- EPSO:
  - Evolutionary PSO
  - Incorporates a selection procedure
  - Self-adapting of parameters

- The particle movement is defined as:

$$V_i^t = w'_{i1} V_i^{t-1} + w'_{i2} r_1 (P_i - X_i^{t-1}) + w'_{i3} r_2 (P'_g - X_i^{t-1})$$

$$X_i^t = X_i^{t-1} + V_i^t$$

# Hybrid PSO : EPSO



- Mutation of weights and global best:

$$w'_{ik} = w_{ik} + \tau.N(0,1)$$

$$P'_g = P_g + \tau'.N(0,1)$$

  - Learning parameters $\tau'$ و $\tau$ can be either fixed or dynamically changing as strategic parameters.

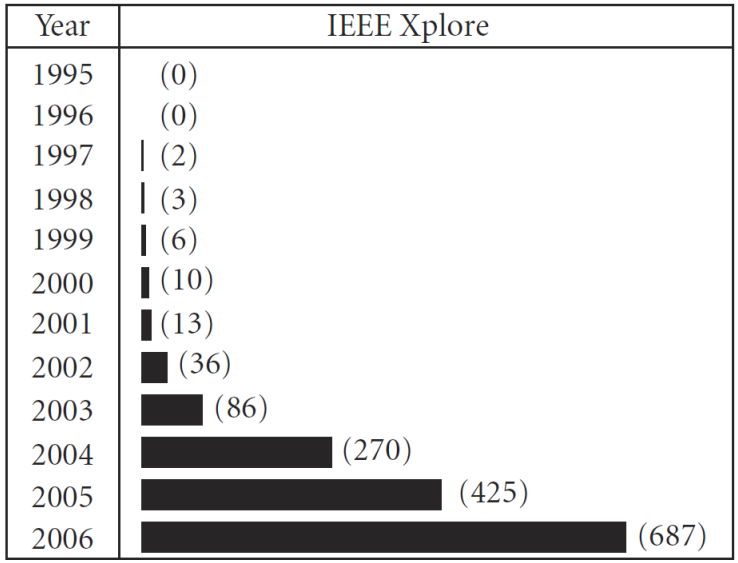
- Survival Selection:
  - Stochastic tournament.

# Simplicity in PSO and Applications



- **Simplicity make PSO be applied in more and more fields**
  - Convenience of realization
  - Properties of low constraint on the continuity of objective function
  - Joint of search space
  - Ability of adapting to dynamic environment
  - ---------

- **Some PSO applications:**
  - Electronics and electromagnetic
  - Signal, Image and video processing
  - Neural networks
  - Communication networks
  - …



# PSO Applications





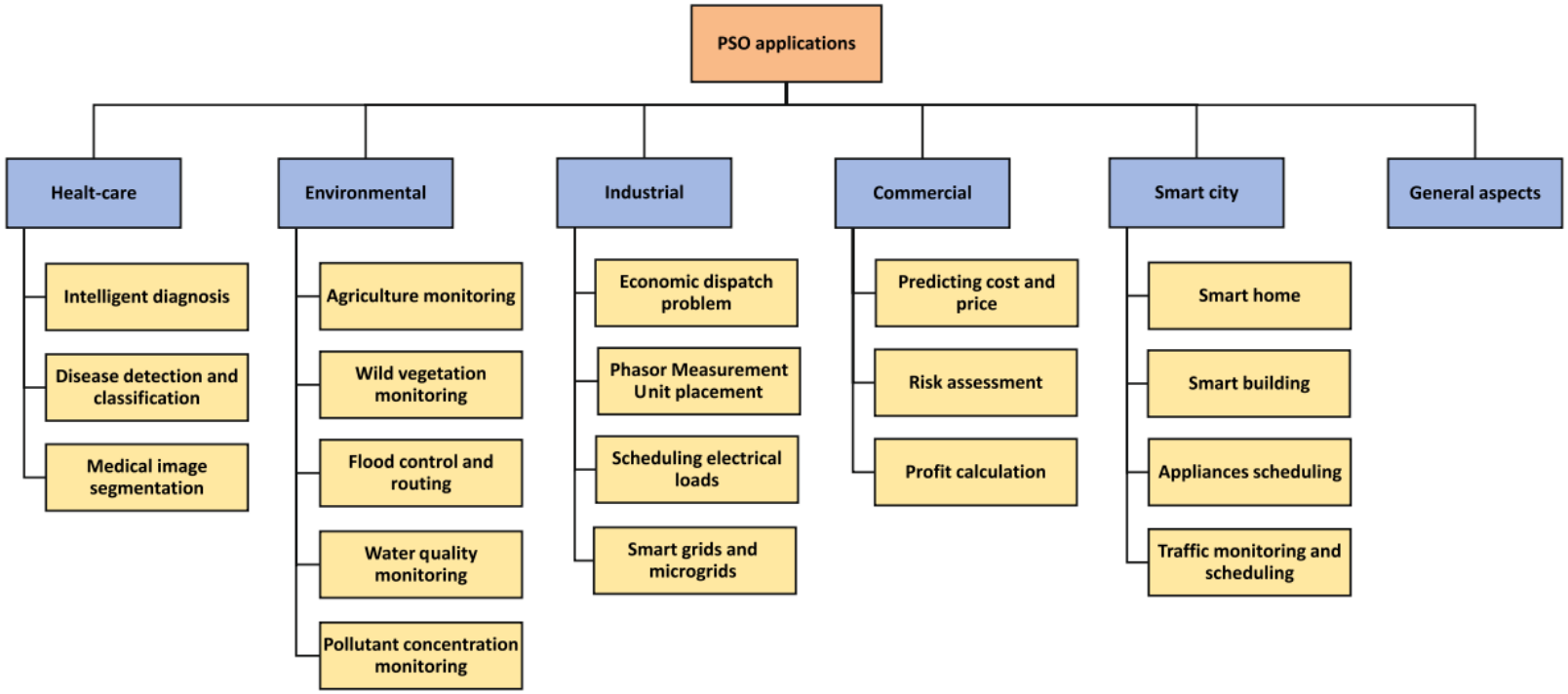
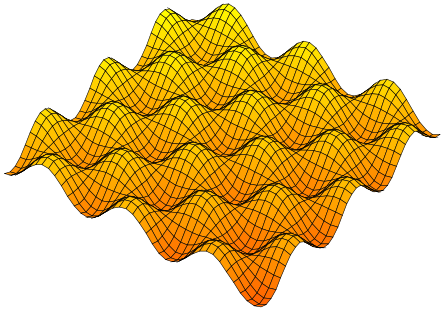



**Fig. 5** The taxonomy of PSO applications

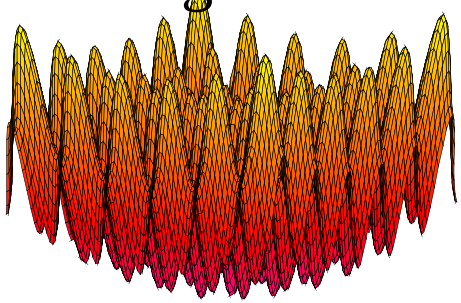
Gad, A.G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. *Arch Computat Methods Eng* **29**, 2531–2561 (2022). https://doi.org/10.1007/s11831-021-09694-4

# Some functions ...

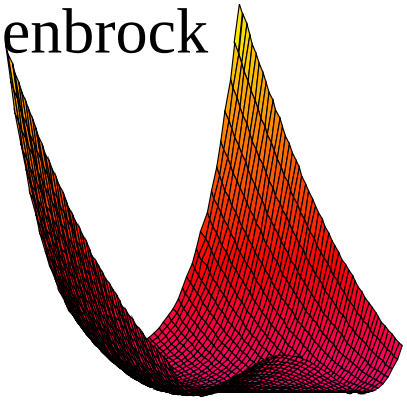
Griewank

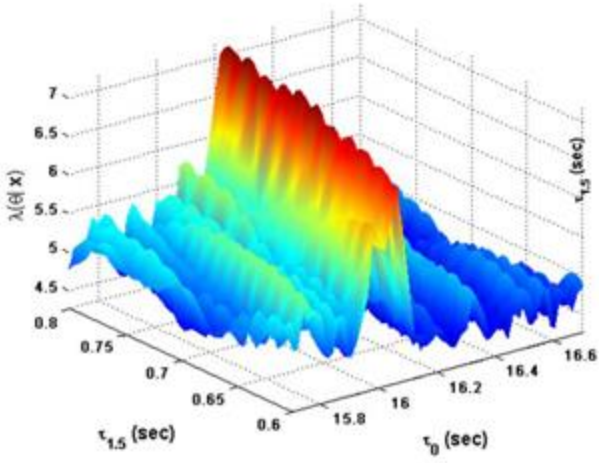


Rastrigin



Rosenbrock





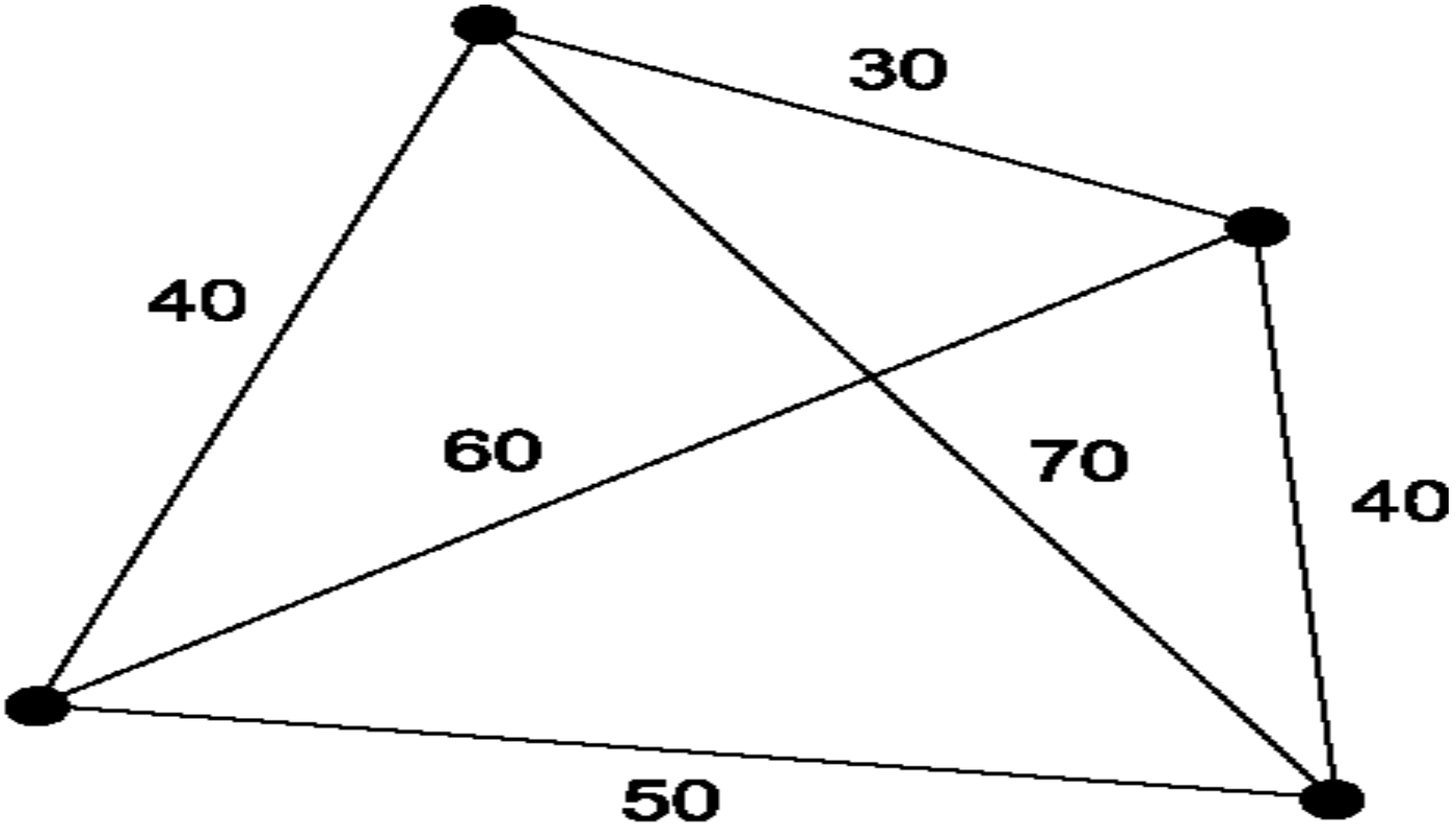
# A Detailed Problem Solving



- [TextBook] 2021 - Evolutionary Optimization Algorithms-CRC Press
- 5.7 Example -> Page 98

# Thanks for your attention

# Ant Colony Optimization to solve Traveling Salesman Problem



# Topics

1. History of Ant Colony Optimization (ACO)
2. Real Ant's Behavior
3. The Concept of Ant System
4. ACO System Concept
5. Pheromone Update Formula
6. Ant Colony Optimization Algorithm
7. Apply Ant Colony Optimization Algorithm to solve Traveling Salesman Problem
8. Example of a Simple AS to solve TSP
9. Application of ACO
10. Advantage and Disadvantages

# History of Ant Colony Optimization

- The first ACO system was introduced by Marco Dorigo in his Ph.D. thesis (1992),
- It was called Ant System (AS).
- AS was initially applied to the traveling salesman problem.
- Applied later to various hard optimization problems

Macro Dorigo

Gambardella

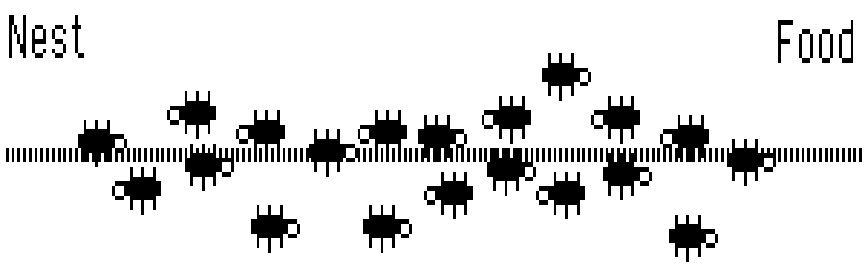
# Real Ant's Behavior

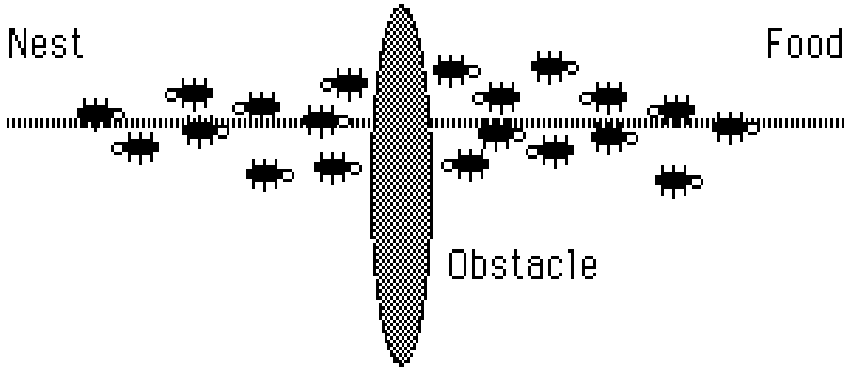
- Natural behavior of ants have inspired scientists to mimic insect operational methods to solve real-life complex problems.

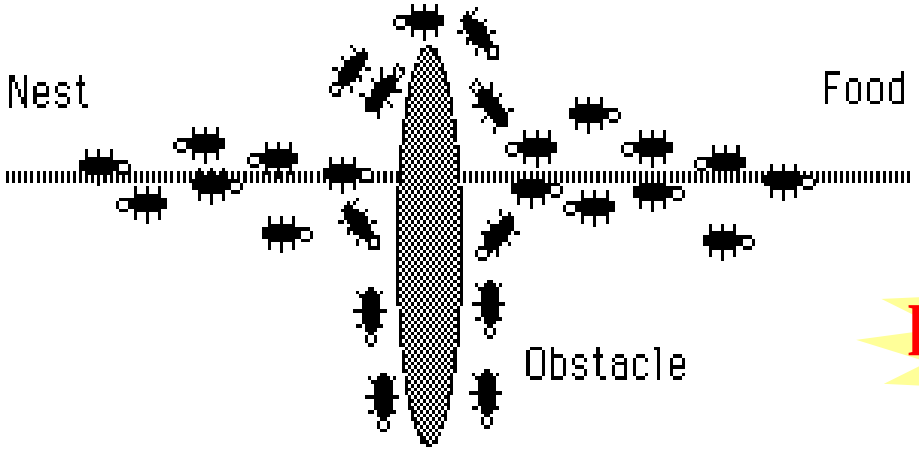
- By observing ant behavior, scientists have begun to understand their means of communication.

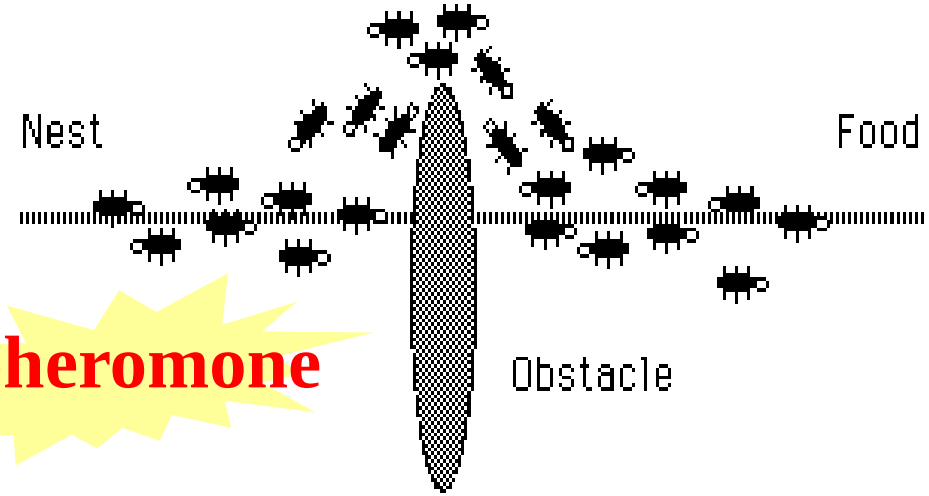
- Ants choose paths depending on pheromone
- After collecting food, paths are marked
- Pheromone accumulation is faster on the shorter path
- After some time, the shortest path has the highest probability

# Real Ant's Behavior



# The Concept of Ant System

- Ants (blind) navigate from nest to food source
- Shortest path is discovered via pheromone trails
  - each ant moves at random
  - pheromone is deposited on path
  - ants detect lead ant's path, inclined to follow
  - more pheromone on path increases the probability of path being followed

- These pheromones evaporate with time.

# The Concept of Ant System

- The shorter path will be reinforced by the pheromones further.
- Finally , the ants arrive at the shortest path.
- Starting node selected at random
- Path selected at random
  - based on amount of “trail” present on possible paths from starting node
  - higher probability for paths with more “trail”
- Ant reaches next node, selects next path
- Continues until reaches starting node
- Finished “tour” is a solution

# The Concept of Ant System

- A completed tour is analyzed for optimality
- “Trail” amount adjusted to favor better solutions
  - better solutions receive more trail
  - worse solutions receive less trail
  - higher probability of ant selecting path that is part of a better-performing tour
- New cycle is performed
- Repeated until most ants select the same tour on every cycle

# Ant Colony Algorithms

- Algorithm was inspired by observation of real ant colonies.
- Ants are essentially blind, deaf and dumb.
- Q: how can ants find the shortest path to food sources?
- Ants deposit *pheromones* on ground that form a trail. The trail attracts other ants.

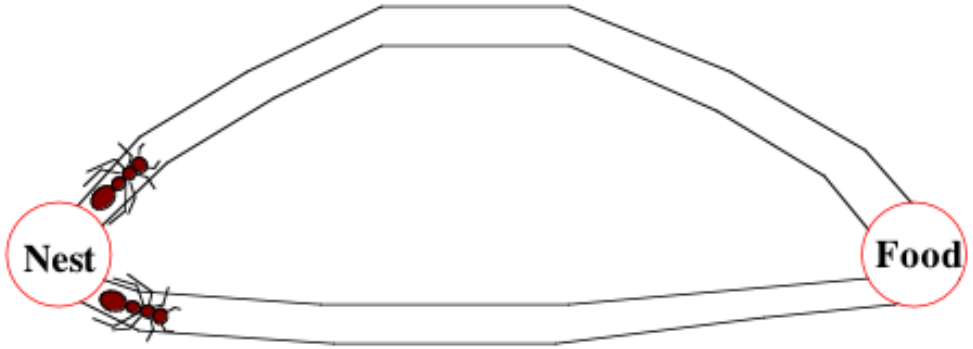
- The ants evaluate the cost of the paths they have traversed.
- The shorter paths will receive a greater deposit of pheromones.
- An evaporation rule will be tied with the pheromones, which will reduce the chance for poor quality solutions.

# How Ants find foods

# How Ants find foods

## Ant foraging

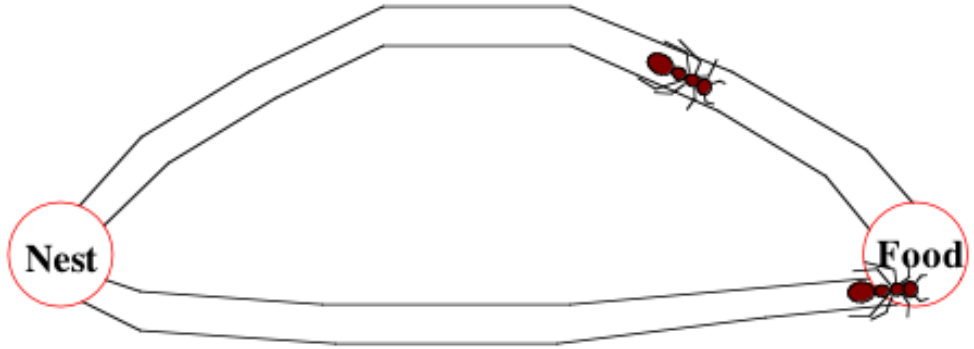
**Cooperative search by pheromone trails**



# How Ants find foods

## Ant foraging

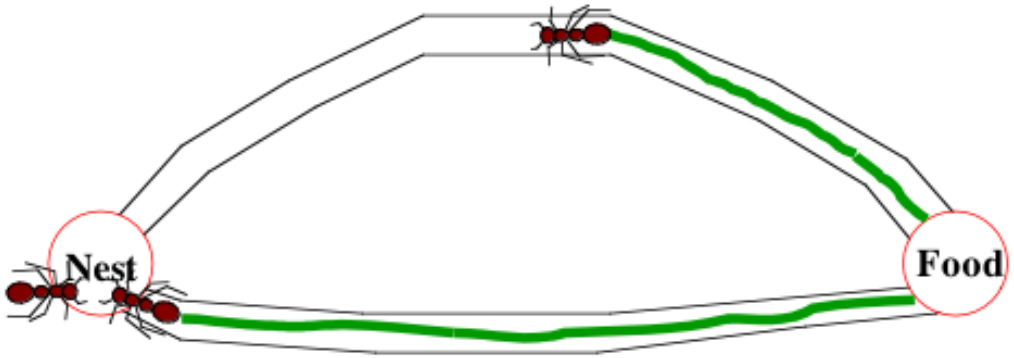
**Cooperative search by pheromone trails**



# How Ants find foods

## Ant foraging

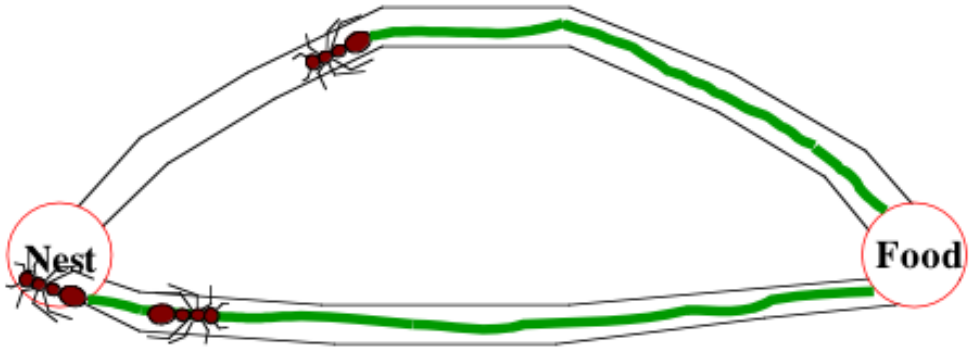
**Cooperative search by pheromone trails**



# How Ants find foods

## Ant foraging

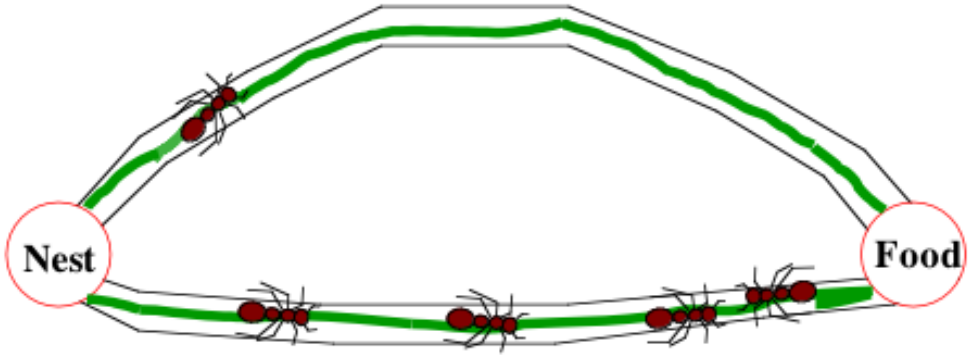
**Cooperative search by pheromone trails**



# How Ants find foods

## Ant foraging

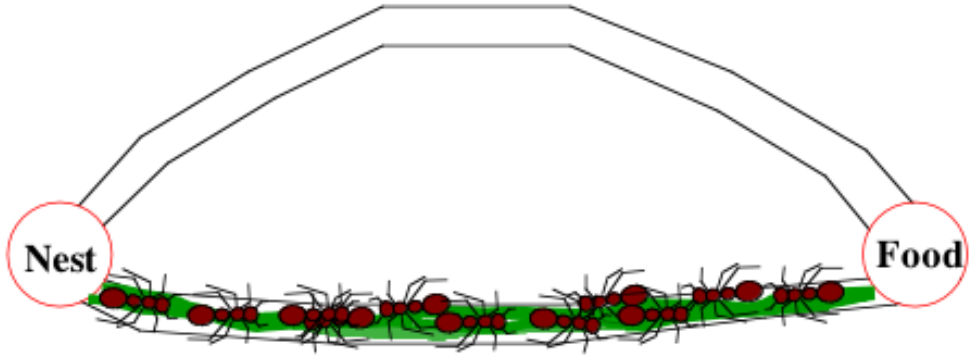
**Cooperative search by pheromone trails**



# How Ants find foods

## Ant foraging

**Cooperative search by pheromone trails**



# Ant System (AS)

- Three different versions are proposed:
  - Ant Density
  - Ant quantity
  - Ant cycle

Pheromone updated after each move of ant from one city to another

Pheromone updates after all ants construct tour

Ant cycle in performed mush better than other two.
We will present Ant-cycle algorithm

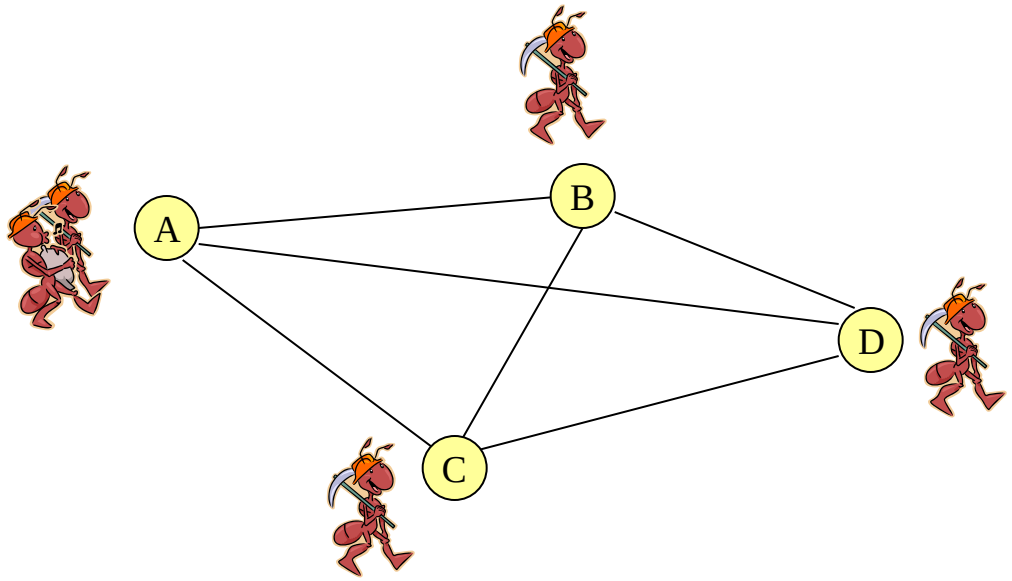
# Ant Systems (AS)

## Ant Systems for TSP:

Graph (N,E): where N = cities/nodes, E = edges

= *the tour cost from city i to city j (edge weight)*

Ant move from one city i to the next j with some transition probability.



# Tour Construction

1. Initially, each ant is put on some randomly chosen city.
2. Tabu list: $N_i^k$ set of all cities that ant k has not visited
3. $n_{ij} = 1/d_{ij}$ , **visibility**: Heuristic desirability of choosing city j when in city i.
4. **Pheromone trail**: $T_{ij}(t)$ This is a global type of information

**Transition probability:** Ant *k*, currently at city *i*, chooses to go to city *j* at *t* th iteration is:

$$P_{ij}^{k}(t) = \frac{[T_{ij}(t)]^{a} \cdot [n_{ij}]^{\beta}}{\sum_{l \in J_i^k} [T_{lk}(t)]^{a} \cdot [n_{lk}]^{\beta}} \quad If\ j \in N_i^k$$

Closest cities are selected based on pheromone and distance.

- α,β determines relative influence of pheromone trail and heuristic information
- If α=0, closest cities are more likely to be selected. Approaches to greedy algorithm
- If β=0, only pheromone amplification is at work

# Pheromone Update Rule

- After all ants constructed their tours, pheromone trails are updated.
- It is done by
  - Firstly, lowering the pheromone strength on all arcs by a constant factor
  - Secondly, allowing each ant to add pheromone on the arcs it has visited.

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

(Trail pheromone decay)

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^{k}(t) \text{ or } Q/L & \text{if arc } (i,j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases}$$

$L^{k}(t)$ is length of *k*th ant's tour. $0 < \rho <= 1$ is pheromone trail evaporation.

- The better the ant's tour is, more pheromone is received by arcs belonging to the tour.
- In general, arcs used by many ants and contained in shorter tours will receive more pheromone.

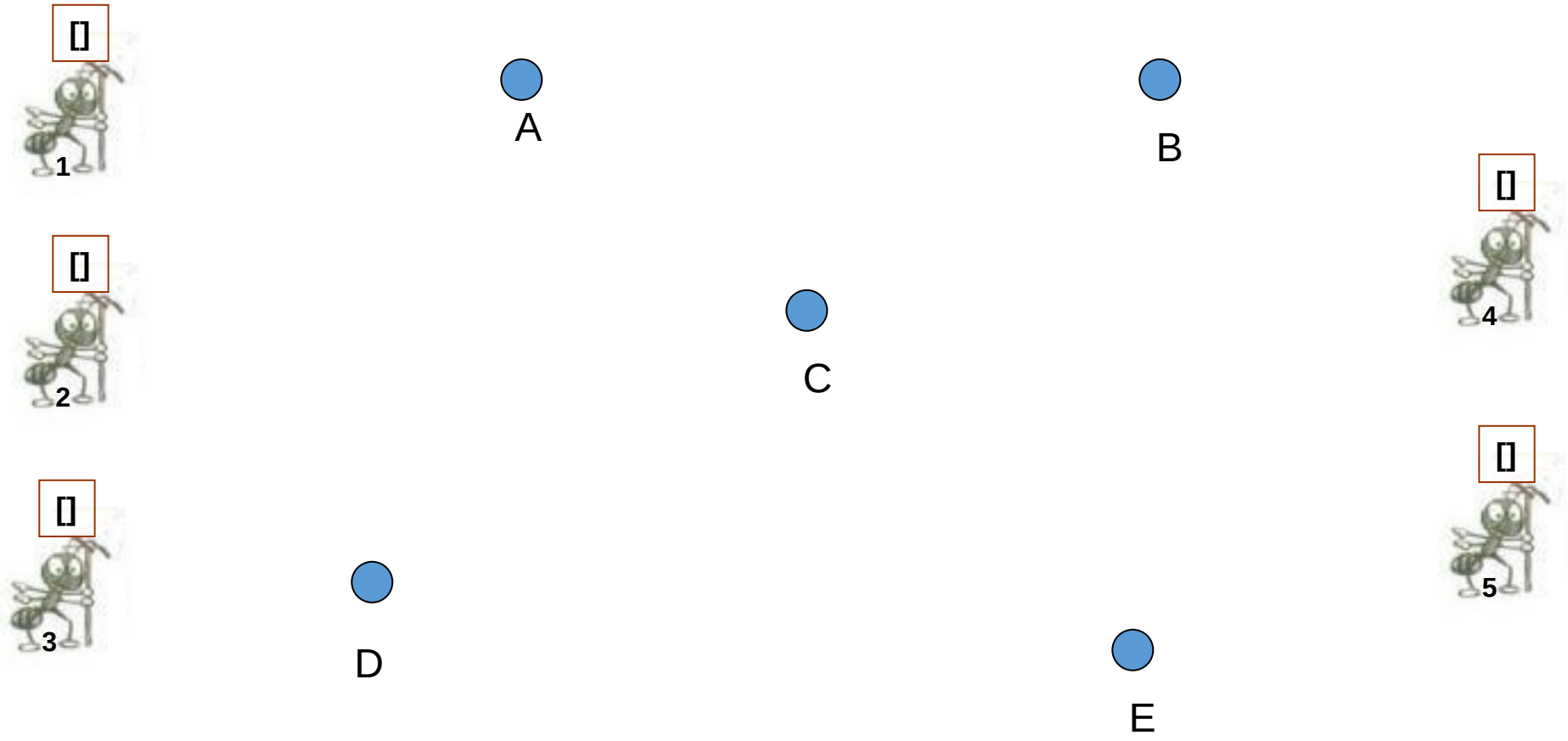
# Key Parameters

$$P_{ij}^{k}(t) = \frac{[T_{ij}(t)]^{a} \cdot [n_{ij}]^{\beta}}{\sum_{l \in J_i^k} [T_{lk}(t)]^{a} \cdot [n_{lk}]^{\beta}}$$

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

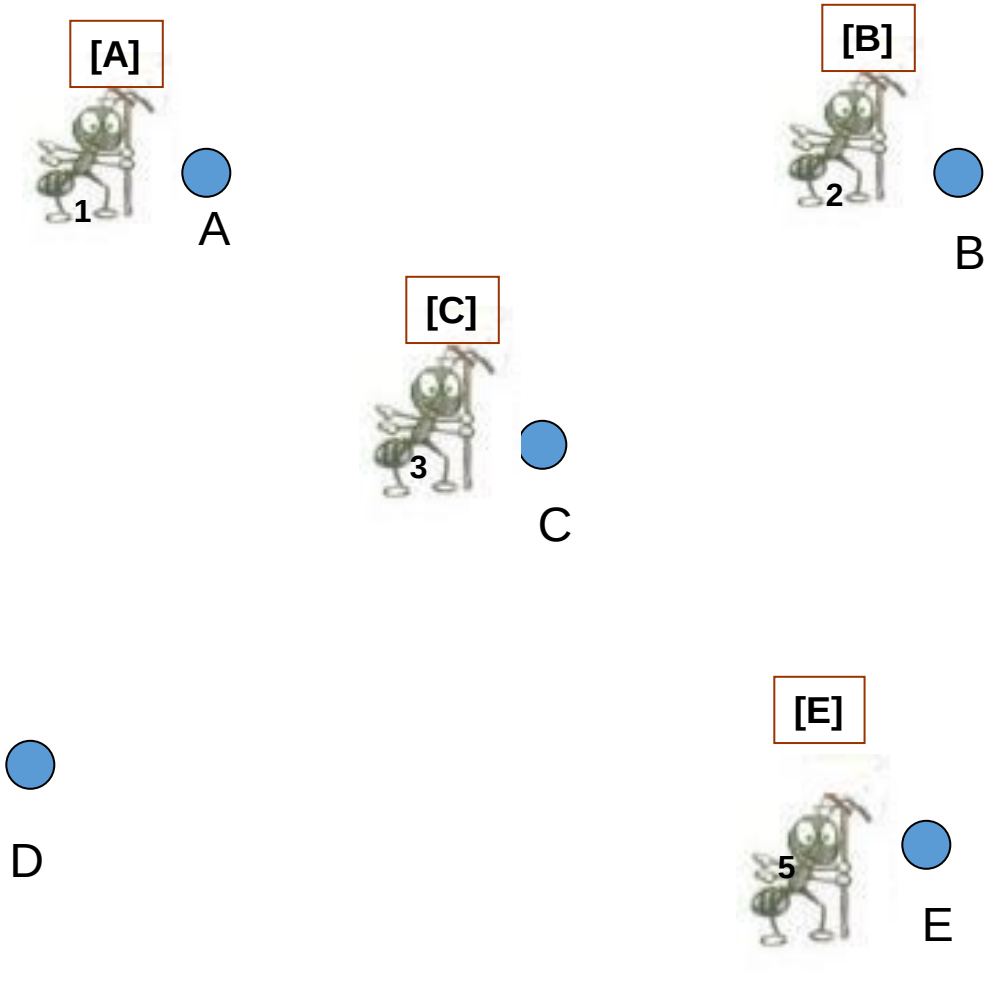
- Trail intensity is given by value of $\tau_{ij}$ which indicates the intensity of the pheromone on the trail segment, (*ij*)
- Trail visibility is $\eta_{ij} = 1/d_{ij}$ : a heuristic function of the desirability of adding edge ($d_{ij}$ is distance from i to j)
- importance of the intensity in the probabilistic transition is $\alpha$
- The importance of the visibility of the trail segment is $\beta$
- The trail persistence or evaporation rate is given as $\rho$
- *Q* is a constant and the amount of pheromone laid on a trail segment employed by an ant (for $\Delta\tau_{ij}^{(k)} \leftarrow Q/L_k$ )
- Initial pheromone is a small amount on all edges

# Example of a Simple AS to solve TSP

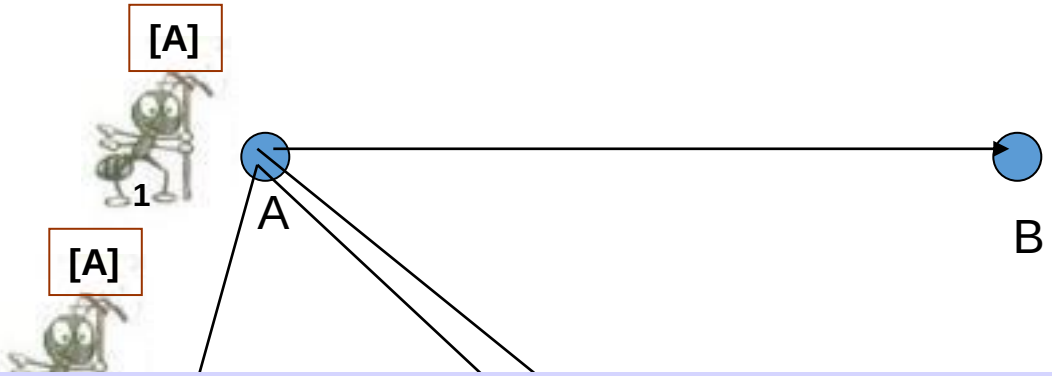



$d_{AB} = 100; d_{BC} = 60\ldots; d_{DE} = 150$

# Example of a Simple AS to solve TSP



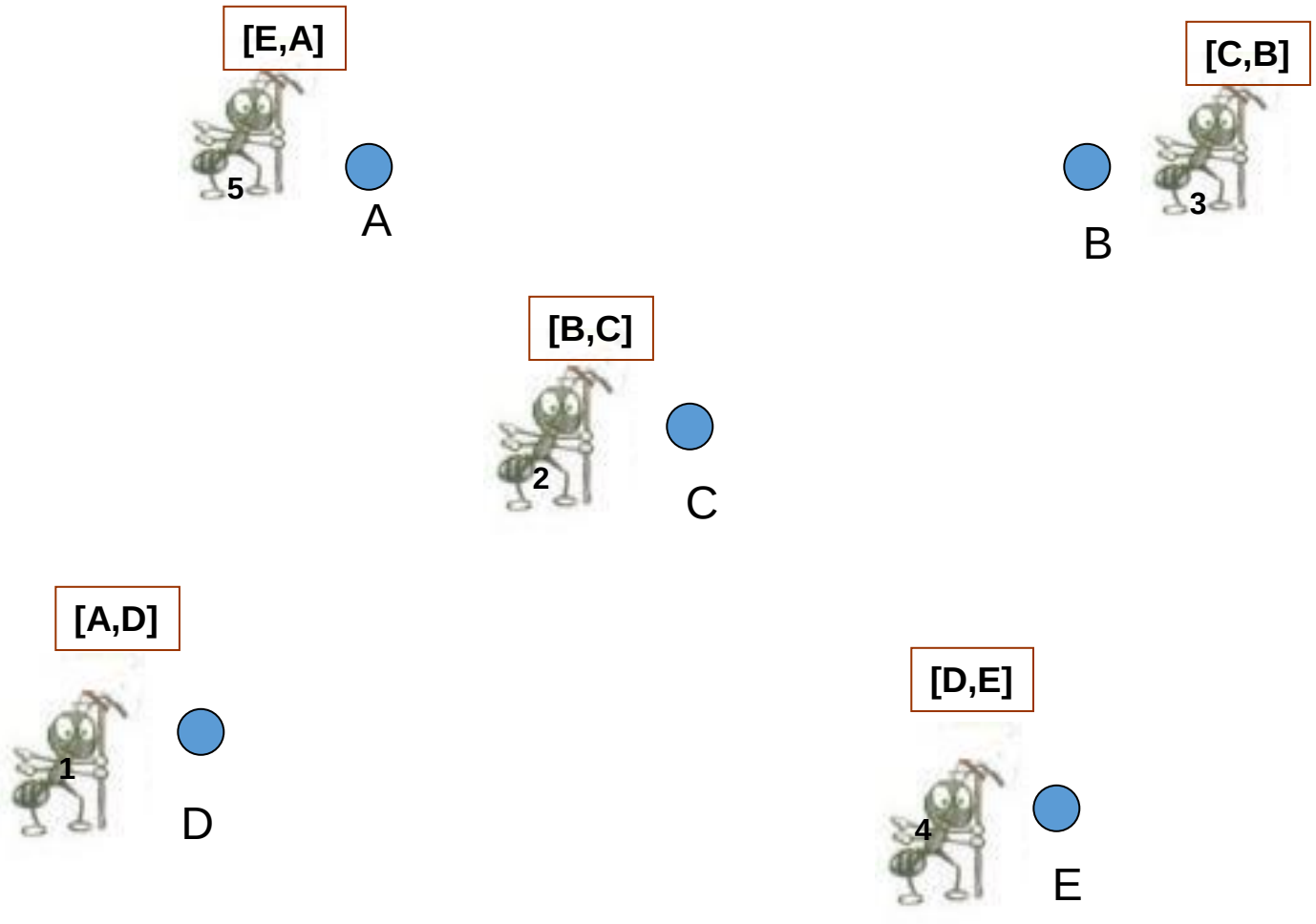
# Example of a Simple AS to solve TSP



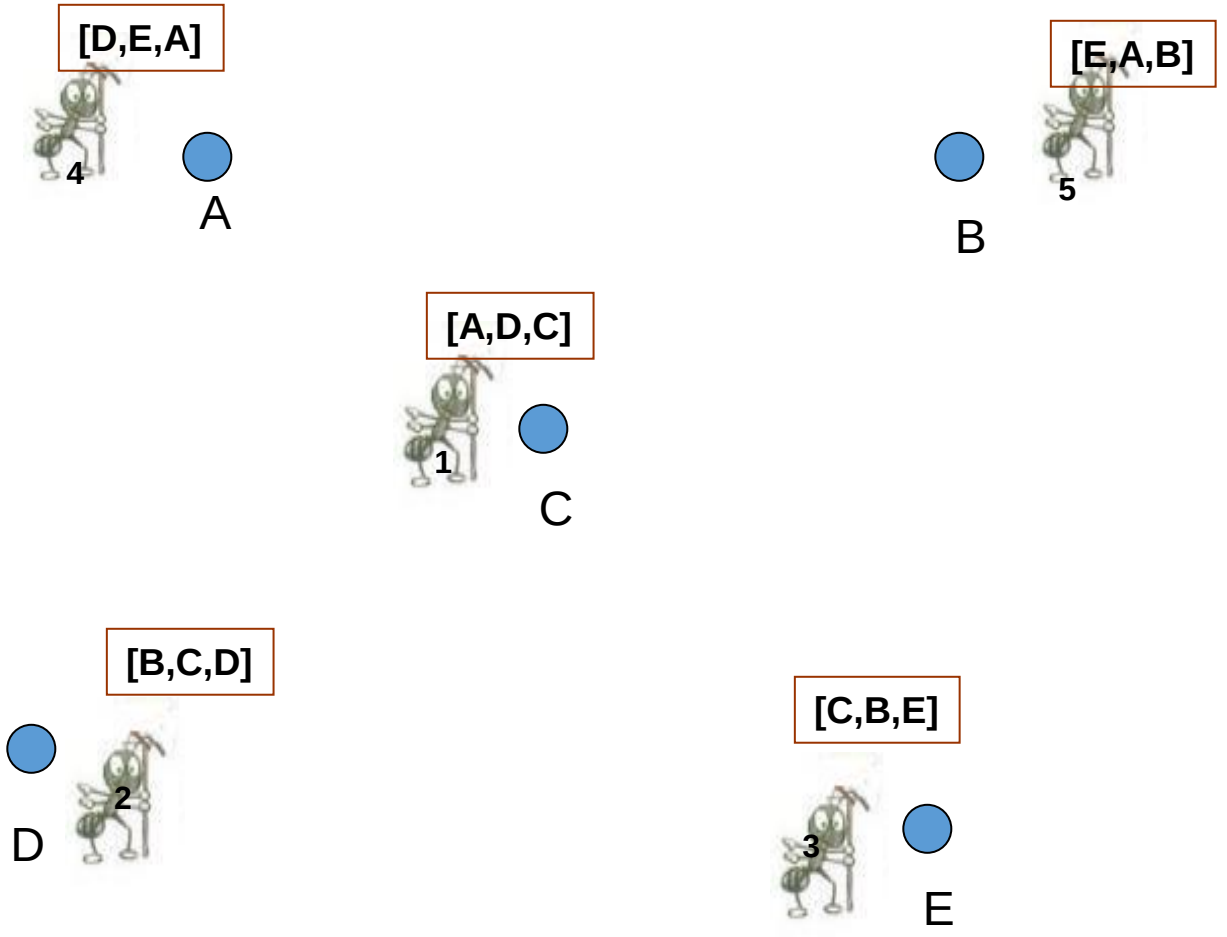

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k \in allowed_k}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases}$$

E

# Example of a Simple AS to solve TSP

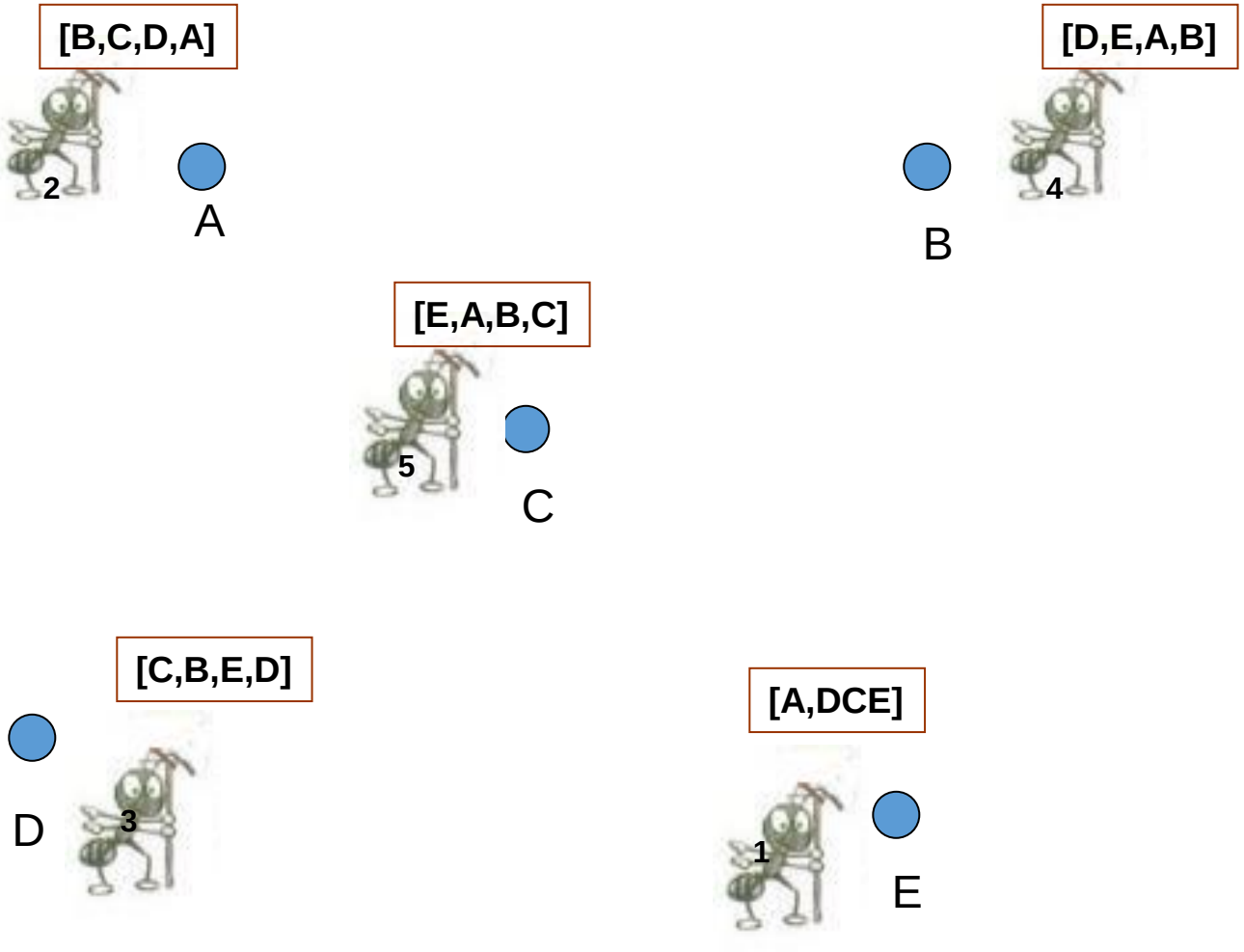


# Example of a Simple AS to solve TSP

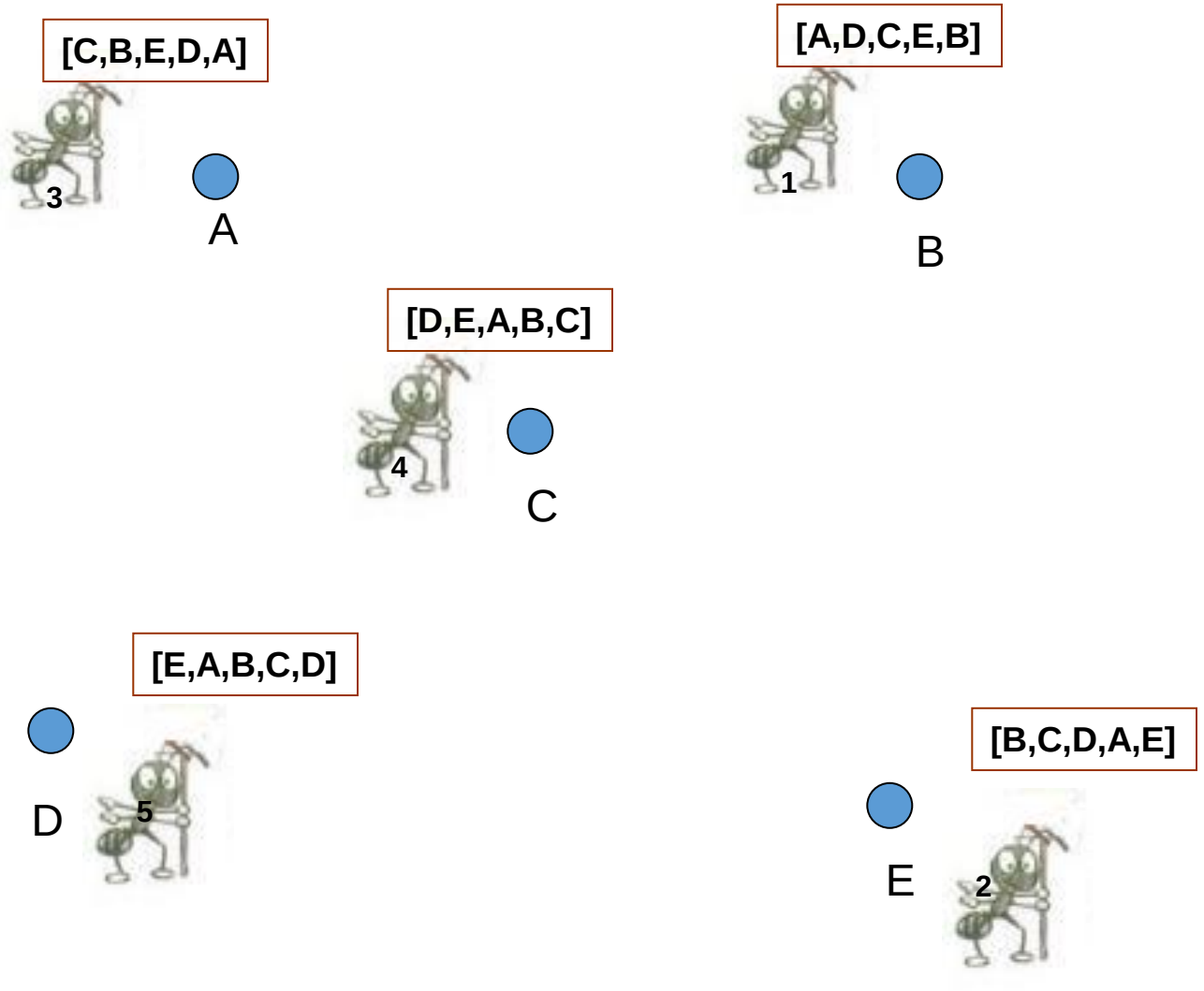



Presented By: Md. Robiul Islam, CSE KUET
(ID: 0907552)

# Example of a Simple AS to solve TSP



# Example of a Simple AS to solve TSP




Presented By: Md. Robiul Islam, CSE KUET
(ID: 0907552)

# Example of a Simple AS to solve TSP

[A,D,C,E,B]


$L_1 = 300$

[B,C,D,A,E]


$L_2 = 450$

$$\Delta\tau_{i,j}^{k} = \begin{cases} \dfrac{Q}{L_k} & if\,(i,j) \in \text{tour} \\ 0 & \text{otherwise} \end{cases}$$

[C,B,E,D,A]

$$\Delta\tau_{A,B}^{total} = \Delta\tau_{A,B}^{1} + \Delta\tau_{A,B}^{2} + \Delta\tau_{A,B}^{3} + \Delta\tau_{A,B}^{4} + \Delta\tau_{A,B}^{5}$$

[D,E,A,B,C]


$L_4 = 280$

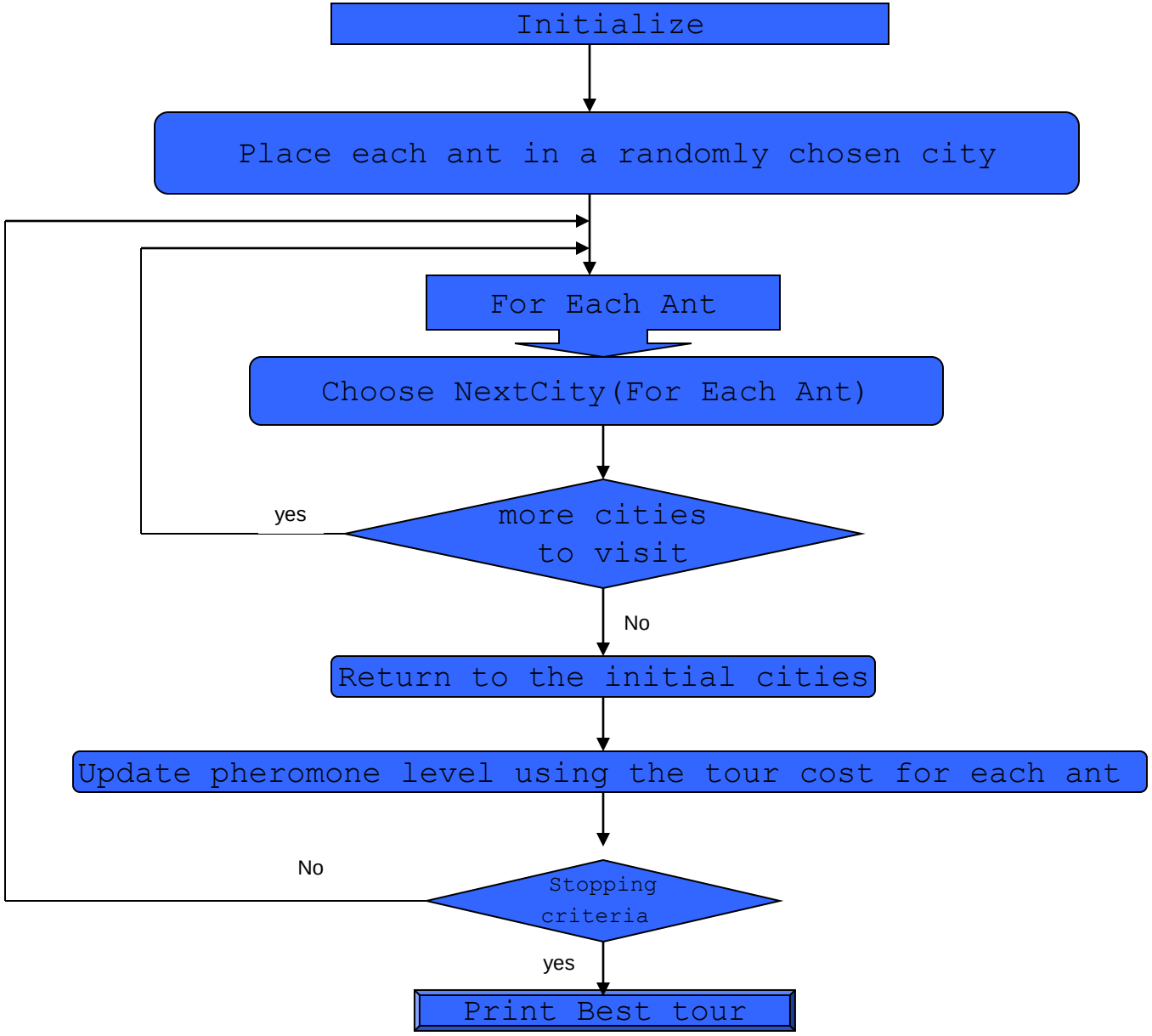
[E,A,B,C,D]


$L_5 = 420$

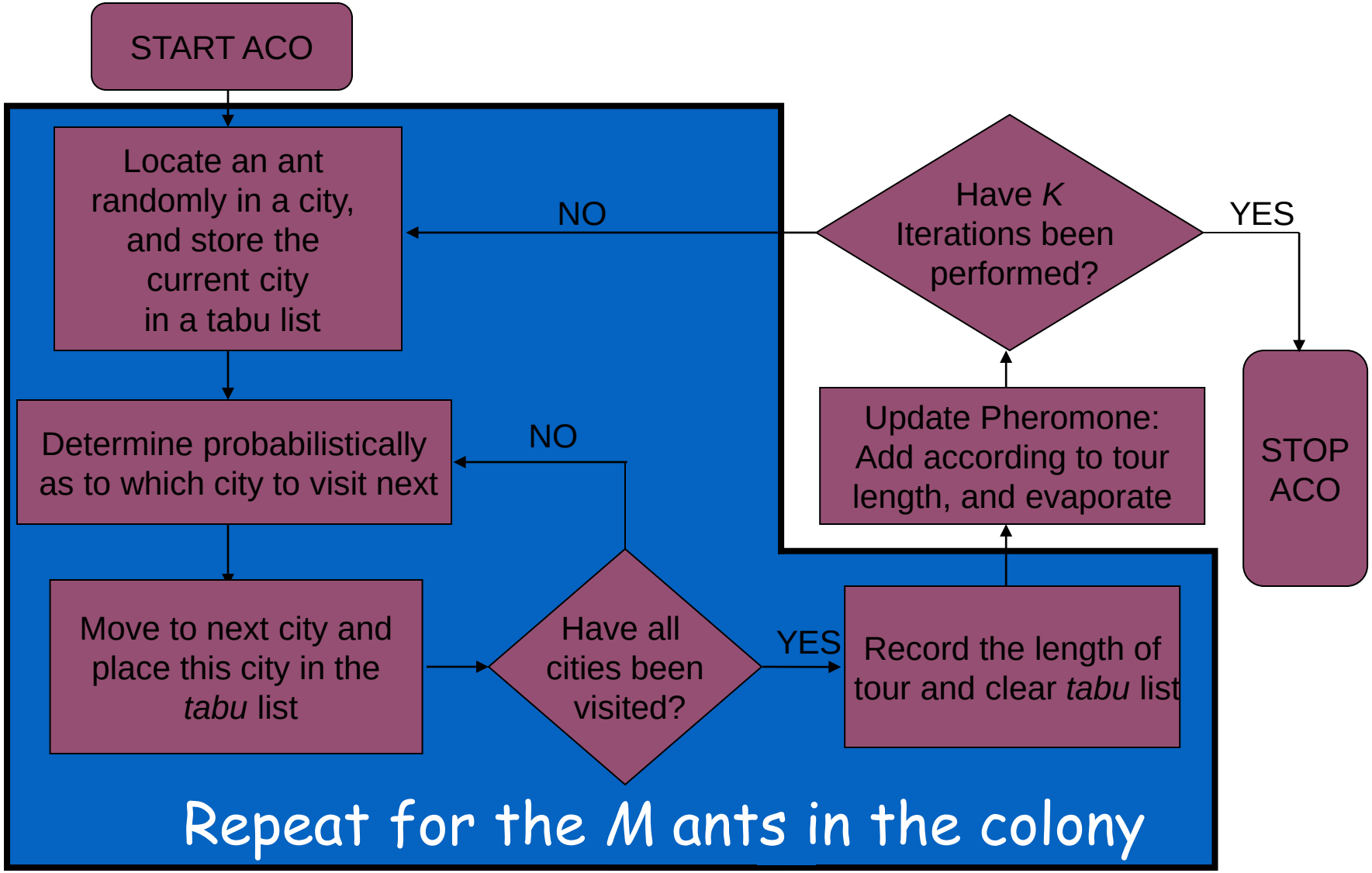
# Application of ACO

- Traveling Salesman Problem (TSP)
- Class Scheduling Problem (CSP)
- Function Optimization
- Vehicle Routing
- Sequential Ordering
- Graph Coloring
- Frequency Assignment
- Train Time Scheduling
- Water Distribution Network
- Quadratic assignment problems (QAP)
- Dynamic routing problems in networks and so on…

# Ant Systems Algorithm for TSP



# Another ACO Flowchart for TSP



# ACO Algorithm for TSP



$n$ = number of cities
$\alpha, \beta$ = relative importance of pheromones vs. heuristic information
$Q$ = deposition constant
$\rho$ = evaporation rate $\in (0, 1)$
$\tau_{ij} = \tau_0$ (initial pheromone between cities $i$ and $j$) for $i \in [1, n]$ and $j \in [1, n]$
$d_{ij}$ = distance between cities $i$ and $j$ for $i \in [1, n]$ and $j \in [1, n]$
While not(termination criterion)
    For $q = 1$ to $n - 1$
        For each ant $k \in [1, N]$
            Initialize the starting city $c_{k1}$ of each ant $k \in [1, N]$
            Initialize the set of cities visited by ant $k$: $C_k \leftarrow \{c_{k1}\}$ for $k \in [1, N]$
            For each city $j \in [1, n]$, $j \notin C_k$
                probability $p_{ij}^{(k)} \leftarrow \left(\tau_{ij}^{\alpha}/d_{ij}^{\beta}\right) / \left(\sum_{m=1, m \notin C_k}^{n} \tau_{im}^{\alpha}/d_{im}^{\beta}\right)$
            Next $j$
            Let ant $k$ go to city $j$ with probability $p_{ij}^{(k)}$
            Use $c_{k,q+1}$ to denote the city selected in the previous line
            $C_k \leftarrow C_k \cup \{c_{k,q+1}\}$
        Next ant
    Next $q$
    $L_k \leftarrow$ total path length constructed by ant $k$, for $k \in [1, N]$
    For each city $i \in [1, n]$ and each city $j \in [1, n]$
        For each ant $k \in [1, N]$
            If ant $k$ went from city $i$ to city $j$
                $\Delta\tau_{ij}^{(k)} \leftarrow Q/L_k$
            else
                $\Delta\tau_{ij}^{(k)} \leftarrow 0$
            End if
        Next ant
        $\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^{N} \Delta\tau_{ij}^{(k)}$
    Next city pair
Next generation

**Figure 10.6** A simple ant system (AS) for solving a TSP. Each generation, some of the pheromone between cities $i$ and $j$ evaporates, but the pheromone also increases due to ants that travel between the two cities.

# ACO Algorithm for TSP



**Algorithm 6.3.1**: Pseudocode for Ant System.

**Input**: ProblemSize, $Population_{size}$, $m$, $\rho$, $\alpha$, $\beta$
**Output**: $P_{best}$

```
1  P_best ← CreateHeuristicSolution(ProblemSize);
2  Pbest_cost ← Cost(S_h);
3  Pheromone ← InitializePheromone(Pbest_cost);
4  while ¬StopCondition() do
5  |   Candidates ← ∅;
6  |   for i = 1 to m do
7  |   |   S_i ← ProbabilisΔτ_ij^(k) ← Q/L_k tructionConstruction(Pheromone,
   |   |   ProblemSize, α, β);
8  |   |   Si_cost ← Cost(S_i);
9  |   |   if Si_cost ≤ Pbest_cost then
10 |   |   |   Pbest_cost ← Si_cost;
11 |   |   |   P_best ← S_i;
12 |   |   end
13 |   |   Candidates ← S_i;
14 |   end
15 |   DecayPheromone(Pheromone, ρ);
16 |   foreach S_i ∈ Candidates do
17 |   |   UpdatePheromone(Pheromone, S_i, Si_cost);
18 |   end
19 end
20 return P_best;
```

# ACO Algorithm for TSP

// Initialize pheromone trails
**for** (every edge i, j) {
$\tau = \tau_0$
}
// Choose the starting town for every ant
**for** ($k = 1; k \leq m; k{+}{+}$) {
Place ant k on a randomly chosen city
}
// Initialize the best tour and length
$T^+$ = the shortest tour found from the beginning
$L^+$ = the length of the best tour
// Main loop
**for** ($t = 1; t \leq Tmax; t{+}{+}$) {
// Compute a tour for every ant
**for** ($k = 1; k \leq m; k{+}{+}$) {
Build tour $T_k(t)$ by applying $n-1$ times the following step:
Choose the next node (city) j with the probability
$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{il \in J_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}(t)]^\beta}$, if $j \in J$
$p_{ij}^k(t) = 0$, if $j \notin J$
where $i$ is the current city.
}
// Compute the tour lengths for all ants
**for** ($k = 1; k \leq m; k{+}{+}$) {
Compute the length $L_k(t)$ of the tour $T_k(t)$ produced by ant $k$
}
// Update the best tour if an improved tour is found
**if** (an improved tour is found) {
Update $T^+$ and $L^+$
**print** $T^+$ and $L^+$
}

// Global update for the pheromone trails
**for** (every edge i, j) {
Update the pheromone trails by applying the rule:
$\tau_{ij}(t) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) + e \cdot \tau_{ij}^e(t)$, where
$\Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij}^k(t)$
$\Delta\tau_{ij}^k(t) = \left\{ \frac{Q/L^k(t), if (i,j) \in T^k(t)}{0, otherwise} \right\}$
and
$\tau_{ij}^e(t) = \left\{ \frac{Q/L^+, if (i,j) \in T^+}{0, otherwise} \right\}$
}
// Calculate the intensity of the pheromone for next iteration
**for** (every edge i, j) {
$\tau_{ij}(t+1) = \tau_{ij}(t)$
}
}

# Advantage and Disadvantages

- For TSPs (Traveling Salesman Problem), relatively efficient
  - for a small number of nodes, TSPs can be solved by exhaustive search
  - for a large number of nodes, TSPs are very computationally difficult to solve (NP-hard) – exponential time to convergence

- Performs better against other global optimization techniques for TSP (neural net, genetic algorithms, simulated annealing)

- Compared to GAs (Genetic Algorithms):
  - retains memory of entire colony instead of previous generation only
  - less affected by poor initial solutions (due to combination of random path selection and colony memory)

# Advantage and Disadvantages

- Can be used in dynamic applications (adapts to changes such as new distances, etc.)
- Has been applied to a wide variety of applications
- As with GAs, good choice for constrained discrete problems (not a gradient-based algorithm)

- Theoretical analysis is difficult:
  - Due to sequences of random decisions (not independent)
  - Probability distribution changes by iteration
  - Research is experimental rather than theoretical
- Convergence is guaranteed, but time to convergence uncertain

# Advantage and Disadvantages

- Tradeoffs in evaluating convergence:
  - In NP-hard problems, need high-quality solutions quickly – focus is on quality of solutions
  - In dynamic network routing problems, need solutions for changing conditions – focus is on effective evaluation of alternative paths

- Coding is somewhat complicated, not straightforward
  - Pheromone “trail” additions/deletions, global updates and local updates
  - Large number of different ACO algorithms to exploit different problem characteristics

# References


[1]. Thomas STÄUTZLE*z* and Marco DORIGO. ACO Algorithms for the Traveling Salesman Problem

[2]. Marco Dorigo. The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, Technical Report IRIDIA-2000-32

[3]. Eric Bonabeau , Marco Dorigo, Guy Theraulaz. Swarm Intelligence - From Natural to Artificial Systems

[4] Prasanna BALAPRAKASH. Ant Colony Optimization under Uncertainty. IRIDIA – Technical Report Series,Technical Report No.TR/IRIDIA/2005-028 November 2005

[5]. Marco Dorigo, *Member, IEEE,* Vittorio Maniezzo and Alberto Colorni. The Ant System:Optimization by a colony of cooperating agents

[6] Michael Guntsch Jürgen Branke. Hand book of Bio-inspired Algorithm and Application: Ant Colony Optimization

[7] Mohamed Belal, Jafaar Gaber, Hoda El-Sayed,Abdullah Almojel. Hand book of Bio-inspired Algorithm and Application: Swarm Intelligence

# References


[8]. Jakob Kierkegaard & Jean-Luc Ngassa. ACO and TSP, 29th of May, 2007

[9]. Jürgen Branke, Michael Stein, Hartmut Schmeck. A Unified View on Metaheuristics and Their Hybridization

[10]. Borut Robiˇc, Peter Korošec, Jurij Šilc. Ant Colonies and the Mesh-Partitioning Problem

[11]. Marco Dorigo, Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, TR/IRIDIA/1996-5

[12]. Marco Dorigo, *Member, IEEE*, Vittorio Maniezzo and Alberto Colorni. The Ant System: Optimization by a colony of cooperating agents

[13]. M. Dorigo and Gianni Di Caro. The Ant colony optimization Meta-heuristic

[14]. Fred Glover, Gari A. Kochenberger. Hand book of Metaheuristic

[15]. Yoshiyuki Nakamichi, Takaya Arita. Diversity control in ant colony optimization

[16]. Jade Herbots, Willy Herroelen,Roel Leus. Experimental Investigation of the Applicability of Ant Colony Optimization Algorithms for Project Scheduling, November 26, 2004

# References

[17]. Mohammed Al-Fayoumi, 2P .Mahanti and 3Soumya Banerjee. OptiTest: Optimizing Test Case Using Hybrid Intelligence. Proceedings of the World Congress on Engineering 2007 Vol I
WCE 2007, July 2 - 4, 2007, London, U.K.

[18]. Dario Floreano and Claudio Mattiussi. Bio-Inspired Artificial Intelligence: THEORIES, METHODS, AND TECHNOLOGIES

[19]. *Pierre Delisle, Michaël Krajecki, Marc Gravel, Caroline Gagné.* PARALLEL IMPLEMENTATION OF AN ANT COLONY OPTIMIZATION METAHEURISTIC WITH OPENMP

[20]. *Vittorio Maniezzo, Luca Maria Gambardella, Fabio de Luigi.* Ant Colony Optimization

[21]. Sorin C. Negulescu, Claudiu V. Kifor, Constantin Oprean. Ant Colony Solving Multiple Constraints Problem: Vehicle Route Allocation. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. III (2008), No. 4, pp. 366-373

[22]. Christian Blum. Review Ant colony optimization: Introduction and recent trends *ALBCOM, LSI, Universitat Politècnica de Catalunya, Jordi Girona 1-3, Campus Nord, 08034 Barcelona, Spain*
Accepted 11 October 2005

# References

[23]. James Lin and David Ansari. Instruction Scheduling – Variation of Max-Min Ant Colony System Optimization Parameters

[24]. Marco Dorigo and Thomas Stützle: Ant Colony Optimization. MIT Press, Cambridge, MA, 2004.

[25]. LinQuan Xie and HongBiao Mei. The Application of the Ant Colony Decision Rule Algorithm on Distributed Data Mining

[26]. Gianni Di Caro. Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks, September 2004

[27]. Saad Ghaleb Yaseen andNada M. A.AL-Slamy. Ant Colony Optimization, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.6, June 2008 351

[28]. T. Stuitzle and H. Hoos. Improving the ant system: a detail report on MAX-MIN ant system

[29]. Caroline Herssens, Amin Mantrach and Marco Saerens. Ant Colony Optimization Revisited from a Randomized Shortest Path Perspective

[30]. Osvaldo G´omez and Benjam´ın Bar´anOmicron ACO. A New Ant Colony Optimization Algorithm

# References


[31]. Gianni A. Di Caro, Frederick Ducatelle and Luca M. Gambardella. Theory and practice of Ant Colony Optimization for routing in dynamic telecommunications networks

[32]. *Vittorio Maniezzo, Luca Maria Gambardella, Fabio de Luigi.* Ant Colony Optimization

[33]. *A. Ketabi, and R. Feuillet.* Ant Colony Search Algorithm for Optimal Generators Start-up during Power System Restoration

[34]. J. Dréo, P. Siarry. Continuous interacting ant colony algorithm based on dense heterarchy

[35]. C. Blum and M. Sample. ACO for FOP shop scheduling: A case study on different pheromone representation

[36]. Jun Zhang, Xiaomin Hu, X. Tan, J.H. Zhong and Q. Huang. Implementation of an Ant Colony Optimization technique for job shop scheduling problem

[37]. Marcin L. Pilat and Tony White. Using Genetic Algorithms to optimize ACS-TSP

[38]. Daniel Angus Ant Colony Optimization: From Biological Inspiration to an Algorithmic Framework, Swinburne University of Technology, Melbourne, Australia, April 21, 2006

# References

[39]. *Vittorio Maniezzo, Anthonilla and Carbonaro.* Ant colony optimization: an overview

[40]. Hozefa M. Botee and Eric Bonabeauy. Evolving Ant Colony Optimization, Santa Fe Institute 1399 Hyde Park Road, Santa Fe, NM 87501, USA

[41]. Hongcheng Zeng, Timo Pukkala, Heli Peltola and Seppo Kellomäki Application of Ant Colony Optimization for the Risk Management of Wind Damage in Forest Planning

[42]. XIAO Jie , ZHOU Ze-kui, ZHANG Guang-xin. Ant colony system algorithm for the optimization of beer fermentation control, *China,* revision accepted Oct. 12, 2003

[43]. Joon-Woo Lee, Jeong-Jung Kim, Byoung-Suk Choi, and Ju-Jang Lee. Improved Ant Colony Optimization Algorithm by Potential Field Concept for Optimal Path Planning

[44]. Bin Yu, ZhongzhenYang, Chuntian Cheng and Chong Liu. OPTIMIZING BUS TRANSIT NETWORK WITH PARALLELANT COLONY ALGORITHM

[45]. Hao Mei, Yantao Tian*, Linan Zu. A Hybrid Ant Colony Optimization Algorithm for Path Planning of Robot in Dynamic Environment1

[46]. Chia-Ho CHEN. A HYBRID ANT COLONY SYSTEM FOR VEHICLE ROUTING PROBLEM WITH TIME WINDOWS, Society for Transportation Studies, Vol. 6, pp. 2822 - 2836, 2005

# References

[47]. Carlos A. Silva and Thomas A. Runkler, Ant Colony Optimization for dynamic Traveling Salesman Problems

[48]. Tony White, Simon Kaegi, Terri Oda, Revisiting Elitism in Ant Colony Optimization

[49]. Stephen Gilmour and Mark Dras. Understanding the Pheromone System within Ant Colony Optimization

[50]. Mesut G¨unes¸, Udo Sorges, Imed Bouazizi. ARA – The Ant-Colony Based Routing Algorithm for MANETs

[51]. Yanjun Li and Tie-Jun WU. A nested Ant Colony Algorithms for hybrid production scheduling

[52]. Ajay C Solai Jawahar . Ant Colony Optimization for Mobile Ad-hoc Networks

[53]. Ryan M. Garlick and Richard S. Barr. Dynamic Wavelength Routing in WDM Networks via Ant Colony Optimization

[54]. Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai. An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment

[55]. Nicolas Durand and Jean-Marc Alliot. Ant Colony Optimization for Air Traffic Conflict Resolution DTI R&D/POM

# References

[56]. Rafael S. Parpinelli, Heitor S. Lopes and Alex A. Freitas.  An Ant Colony Algorithm for Classification Rule Discovery

[57]. Thomas Weise. Global Optimization Algorithm Version: 2009-06-26

[58]. Jamaludin sallimr, wan Muhammad Syahrir wan Hussin, Rosni Abdullah, Ahamad Tajudin Abdul Khadera. A Background Study on Ant Colony Optimization Metaheuristic and its Application Principles in Resolving Three Combinatorial Optimization Problems

[59]. In`es Alaya, Christine Solnon and Khaled Gh´edira. ANT ALGORITHM FOR THE MULTIDIMENSIONAL KNAPSACK PROBLEM

[60]. Ehsan Salari and Kourosh Eshghi, An ACO Algorithm for the Graph Coloring Problem Int. J. Contemp. Math. Sciences, Vol. 3, 2008, no. 6, 293 - 304

[61]. Rafael S. Parpinelli1, Heitor S. Lopes, and Alex A. Freitas. Data Mining with an Ant Colony Optimization Algorithm

[62]. Fangqing Zhao1, Fanggeng Zhao2, Tao Li1 and Donald A. Bryant. A new pheromone trail-based genetic algorithm for comparative genome assembly Published online 29 April 2008 Nucleic Acids Research, 2008, Vol. 36, No. 10 3455–3462

# References

[63]. Mauro Birattari¤ Prasanna Balaprakash¤ Marco Dorigo. ACO/F-Race: Ant Colony Optimization and Racing Techniques for Combinatorial Optimization Under Uncertainty, MIC2005.

[64]. Jeong-Jun Suh, Shan Guo Quan, Seung-Hoon Park, and Young Yong Kim. Adaptive File Distribution in P2P Network Using Ant Colony Optimization for Smart Home Environment

[65]. MARCO DORIGO and THOMAS ST¨UTZLE. An Experimental Study of the Simple Ant Colony Optimization Algorithm

[66]. Veronica Lopez, Jose A. Gamez, and Luis delaOssa. Improvement of a car racing controller by means of Ant Colony Optimization algorithms

[67]. *Mostafa Abd-El-Barr, Sadiq M. Sait, Bambang A. B. Sarif, Uthman Al-Saiari.* COMBINATIONAL LOGIC CIRCUITS DESIGN THROUGH ANT COLONY OPTIMIZATION ALGORITHM

[68]. Xingguo Chen, Hao Wang, Weiwei Wang, Yinghuan Shi, and Yang Gao.Apply Ant Colony Optimization to Tetris

[69]. Wong, L.-H., & Looi. Adaptable Learning Pathway Generation with Ant Colony Optimization

Wong, L.-H., & Looi, C.-K. (2009).

# References

[63]. Mauro Birattari¤ Prasanna Balaprakash¤ Marco Dorigo. ACO/F-Race: Ant Colony Optimization and Racing Techniques for Combinatorial Optimization Under Uncertainty, MIC2005.

[64]. Jeong-Jun Suh, Shan Guo Quan, Seung-Hoon Park, and Young Yong Kim. Adaptive File Distribution in P2P Network Using Ant Colony Optimization for Smart Home Environment

[65]. MARCO DORIGO and THOMAS ST¨UTZLE. An Experimental Study of the Simple Ant Colony Optimization Algorithm

[66]. Veronica Lopez, Jose A. Gamez, and Luis delaOssa. Improvement of a car racing controller by means of Ant Colony Optimization algorithms

[67]. *Mostafa Abd-El-Barr, Sadiq M. Sait, Bambang A. B. Sarif, Uthman Al-Saiari.* COMBINATIONAL LOGIC CIRCUITS DESIGN THROUGH ANT COLONY OPTIMIZATION ALGORITHM

[68]. Xingguo Chen, Hao Wang, Weiwei Wang, Yinghuan Shi, and Yang Gao.Apply Ant Colony Optimization to Tetris

[69]. Wong, L.-H., & Looi. Adaptable Learning Pathway Generation with Ant Colony Optimization

Wong, L.-H., & Looi, C.-K. (2009).

# References


[70]. Kwang Mong Sim and Weng Hong Sun, *Member, IEEE.* Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions

[71]. Hua Chen and Albert M. K. Cheng. Applying Ant Colony Optimization to the Partitioned Scheduling Problem for Heterogeneous Multiprocessors

[72]. Cristian Martinez. An ACO algorithm for image compression, October 29, 2006

[73]. Aruna A. Priyanto, Adiwijaya, W. Maharani. IMPLEMENTATION OF ANT COLONY OPTIMIZATION ALGORITHM ON THE PROJECT RESOURCE SCHEDULING PROBLEM

[74]. Philip Christian C. Zuniga, Maia Malonzo, Henry Adorna and Prospero Naval. An Ant Colony Optimization Algorithm for the Network Inference and Parameter Estimation of S-Systems International Conference on Molecular Systems Biology 2009

[75]. Mehrabi, Saeed Mehrabi† and Abbas Mehrabi. Ali D. A PRUNING BASED ANT COLONY ALGORITHM FOR MINIMUM VERTEX COVER PROBLEM

[76]. Marc REIMANN ANALYZING A VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS USING ANT COLONY OPTIMIZATION

# References

[77]. P. Jaganathan1*, K.Thangavel2 A. Pethalakshmi3, M. Karnan4 -Classification Rule Discovery with Ant Colony Optimization and Improved Quick Reduct Algorithm

[78]. Vittorio Maniezzo, Matteo Roffilli. VERY STRONGLY CONSTRAINED PROBLEMS:
AN ANT COLONY OPTIMIZATION APPROACH

[79]. Chung-wen Chiang, Yi-chan Lee, Chungnan Lee and Ta-yuan Chou. Ant Colony Optimization for Task Matching and Scheduling

[80]. Joao Pedro, Joao Pires, and Joao Paulo Carvalho . Ant Colony Optimization for Distributed Routing Path Optimization in Optical Burst-Switched Networks

[81]. Pan Junjie and Wang Dingwei. An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem

[82]. Carlos M. Fernandes, Agostinho C. Rosa and Vitorino Ramos. Binary Ant Algorithm

[83]. Malika Bessedik, Rafik Laib, Aissa Boulmerka et Habiba Drias. Ant Colony System for Graph Coloring Problem

# References

[84]. TAN Guan-Zheng1;3 HE Huan2 SLOMAN Aaron. Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots

[85]. Sofie Demeyer,* Marc De Leenheer, Jurgen Baert, Mario Pickavet, and Piet Demeester Ant colony optimization for the routing of jobs in optical grid networks

[86]. Ying-Shiuan You. Parallel Ant System for Traveling Salesman Problem on GPUs

[87]. Nada M. A. Al Salami. Ant Colony Optimization Algorithm

[88]. Holger R. Maier1; Angus R. Simpson2; Aaron C. Zecchin3; Wai Kuan Foong4; Kuang Yeow Phang5;Hsin. Ant Colony Optimization for Design of Water Distribution Systems

[89]. Karl O. Jones, André Bouffet. COMPARISON OF BEES ALGORITHM, ANT COLONY OPTIMISATION AND PARTICLE SWARM OPTIMISATION FOR PID CONTROLLER TUNING

[90]. Farzaneh Jalalinejad, Farhang Jalali-Farahania, Navid Mostoufi, Rahmat Sotudeh-Gharebagh Ant Colony Optimization: A Leading Algorithm in Future Optimization of Chemical Processes

# Artificial Bee Colony (ABC) Optimization Algorithm

Dr. Md. Aminul Haque Akhand
Dept. of CSE, KUET

Acknowledgements:

- C. C. Hung et al. - *School of Computing and Software Engineering, Southern Polytechnic State University, Marietta, Georgia USA*
- *Ziad Salem - Computer Engineering Department, Aleppo University, Syrian Arab Republic*

# Contents

- Bees in Nature
- Motivation to Optimization Algorithm
- Artificial Bee Colony (ABC) Optimization
- Problem Solving with ABC

# Intelligent Behaviors of Natural Bees

**1. The Power of Bee Democracy**

**https://www.youtube.com/watch?v=NDnQ4pAjBUg**

**Short Description: Bee in Nature, Bee and Neuron Relation and ABC**

**2. Facts About Bees 🐝 - Secret Nature | Bee Documentary | Natural History Channel**

**https://www.youtube.com/watch?v=mZTLatV1YIQ**

**N.B.: Information regarding ABC after 35 Mins**

**3. How Do Honeybees Get Their Jobs? | National Geographic**

**https://www.youtube.com/watch?v=9ePic3dtykk**

# Motivation to ABC

- There was a great interest between researchers to generate search algorithms that find near-optimal solutions in reasonable running time
- The Swarm Intelligence (SI)-based Algorithm is a search algorithm capable of locating good solutions efficiently
- The algorithm could be considered as belonging to the category of “Intelligent Optimisation Tools”
- **SIAs** mince nature’s methods to derive a search towards the optimal solution
- The key difference between **SIAs** and **direct search** algorithms such as **Hill Climbing** is that SIAs use a population of solutions for every iteration instead of a single solution
- As a population of solutions is processed in an iteration, the outcome of each iteration is also a population of solutions

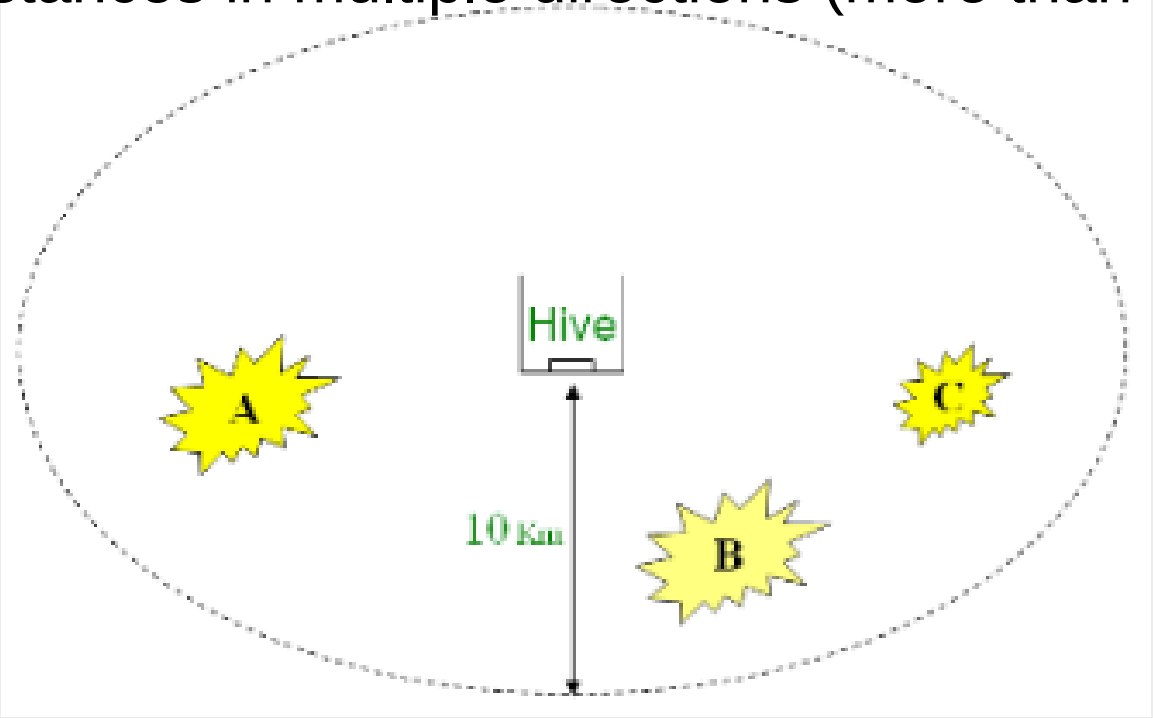
# Motivation to ABC

- **SIAs** include:
  - *The **Ant Colony Optimisation** (ACO) algorithm*
  - *The **Genetic Algorithm** (GA)*
  - *The **Particle Swarm Optimisation** (PSO) algorithm*
  - *Others……(**Bee Algorithm**)*

Artificial Bee Colony (ABC) Algorithm is main concern of this Lecture

# Points from Bees in Nature

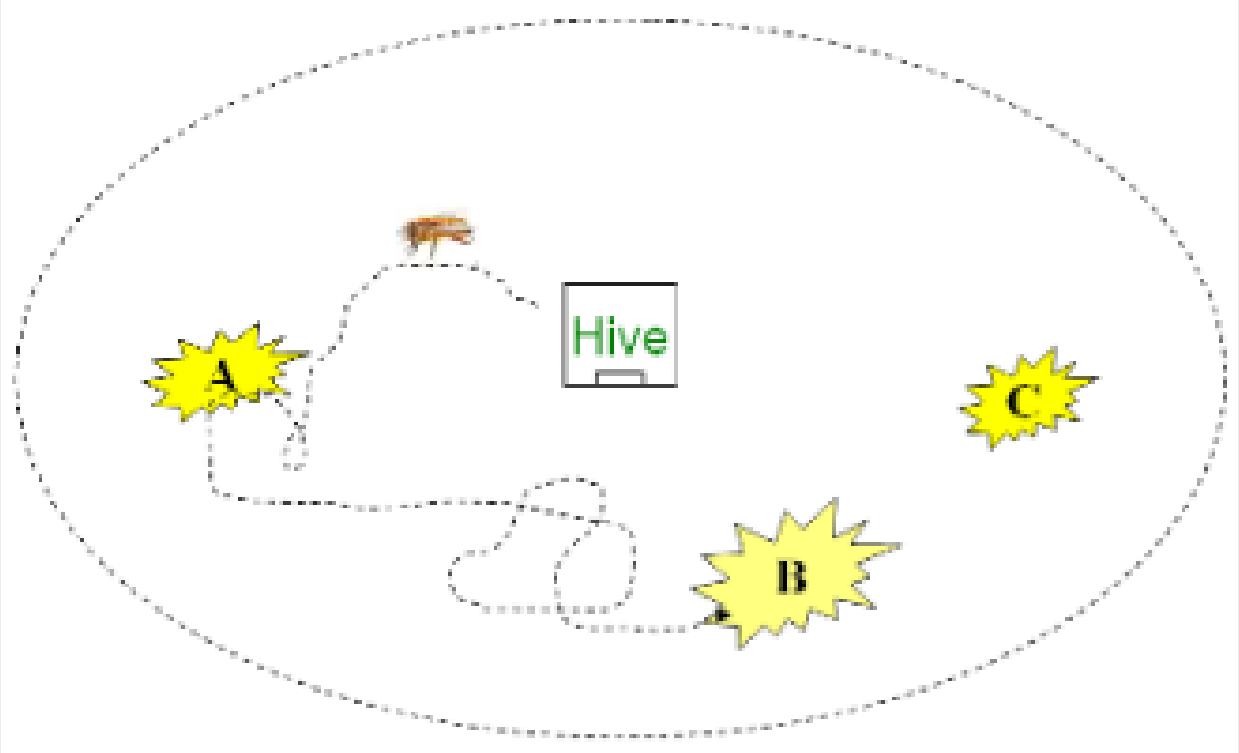
1- **A colony of honey bees** can extend itself over long distances in multiple directions (more than 10 km)




Flower patches with plentiful amounts of nectar or pollen that can be collected with less effort should be visited by more bees, whereas patches with less nectar or pollen should receive fewer bees

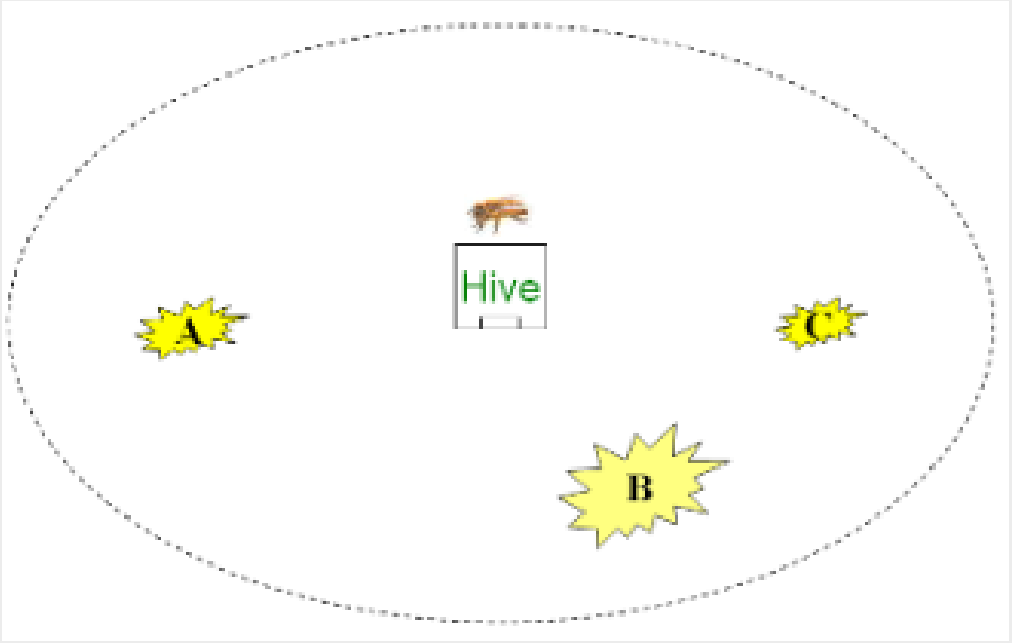
# Points from Bees in Nature

2- **Scout bees** search randomly from one patch to another



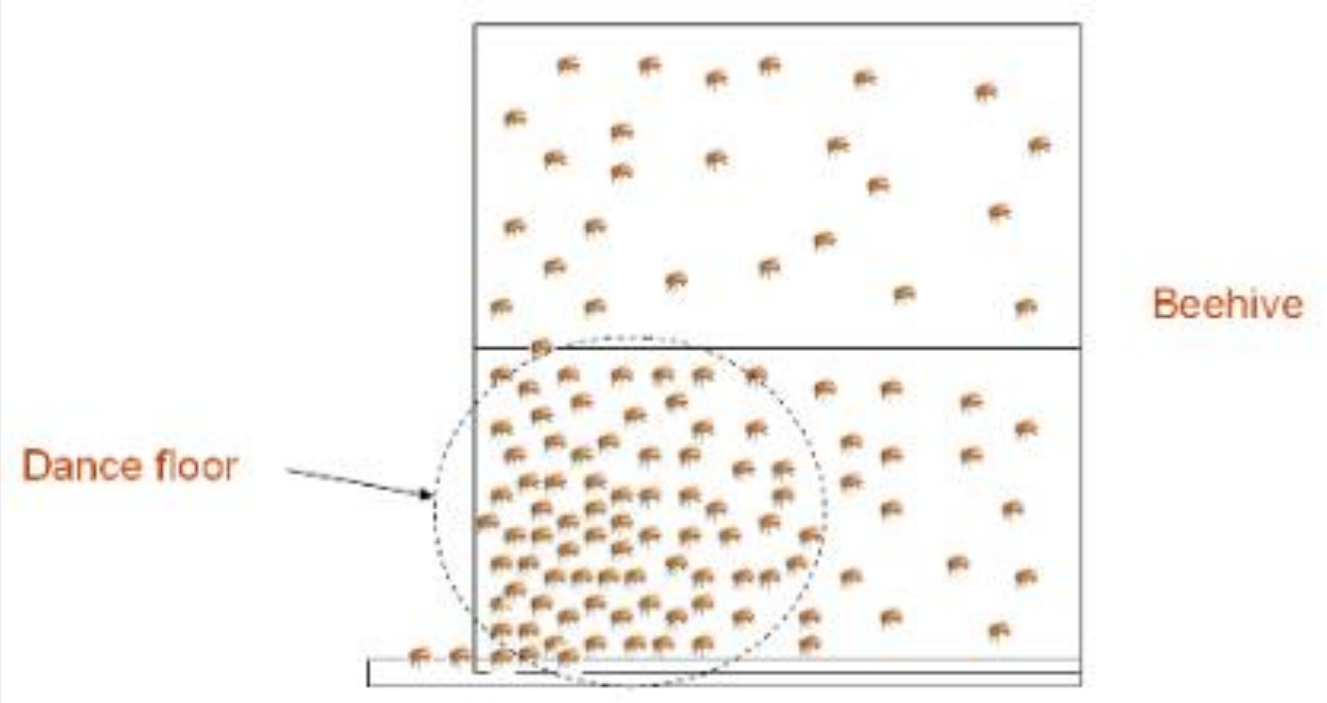
# Points from Bees in Nature

3- The bees who return to the hive, **evaluate** the different patches depending on certain quality threshold (measured as a combination of some elements, such as sugar content)



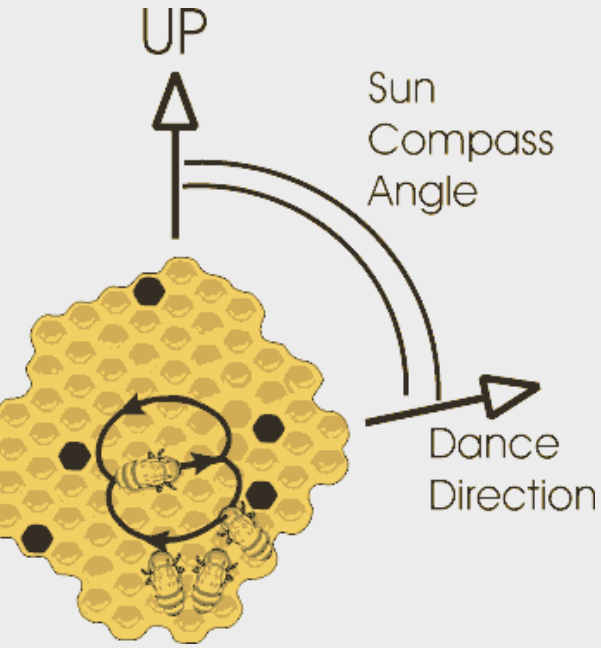
# Points from Bees in Nature

4- They deposit their nectar or pollen go to the "**dance floor**" to perform a "waggle dance"



# Points from Bees in Nature

**5-** Bees communicate through the waggle dance which contains the following information:




1. The **direction** of flower patches (angle between the sun and the patch)
2. The **distance** from the hive (duration of the dance)
3. The **quality** rating (fitness) (frequency of the dance)

These information helps the colony to send its bees precisely

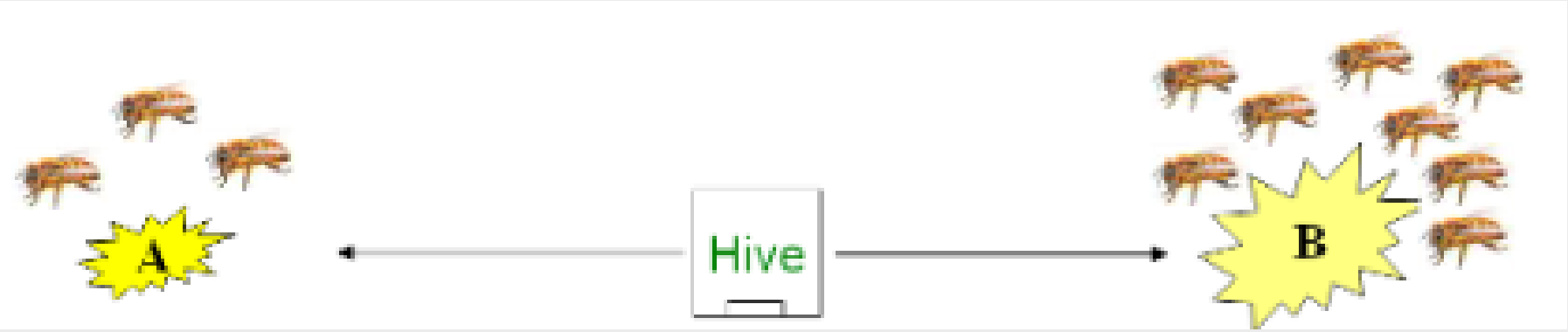
6- **Follower bees** go after the dancer bee to the patch to gather food efficiently and quickly

# Points from Bees in Nature

7- The same patch will be **advertised** in the waggle dance again when returning to the hive is it still good enough as a food source (**depending on the food level**) and more bees will be recruited to that source

8- More bees visit flower patches with plentiful amounts of nectar or pollen

Thus, according to the fitness, patches can be visited by more bees or may be abandoned



# Artificial Bee Colony (ABC) Algorithm

- ABC Algorithm is a population-based search algorithm inspired by the natural foraging behaviour of honey bees to find the optimal solution.
- The algorithm performs a kind of neighbourhood search combined with random search.

# ABC Algorithm (Cont.)

Three type of Bees in ABC:

1) Employed bees

2) Onlooker bees, and

3) Scouts.

Employed and onlooker bees perform the exploitation search.

Scouts carry out the exploration search.

# ABC Algorithm (Cont.)

ABC employs four different selection processes:

1) a global selection process used by onlookers,

2) a local selection process carried out in a region by employed and onlooker bees,

3) a greedy selection process used by all bees, and

4) a random selection process used by scouts.

# ABC Algorithm (Cont.)

## The ABC algorithm Steps:

Step 1: Initialize by picking *k* random **Employed** bees from data.

Step 2: Send **Scout** bees and test against Employed bees

(replace if better than Employed is found).

Step 3: Send Onlooker bees to Employed.

Step 4: Test Onlooker bees against Employed

(replace if better than Employed is found).

Step 5: Reduce the radius of Onlooker bees.

Step 6: Repeat steps 2 to 5 for a given number of iterations.

# ABC Algorithm (Cont.): Pseudo Code

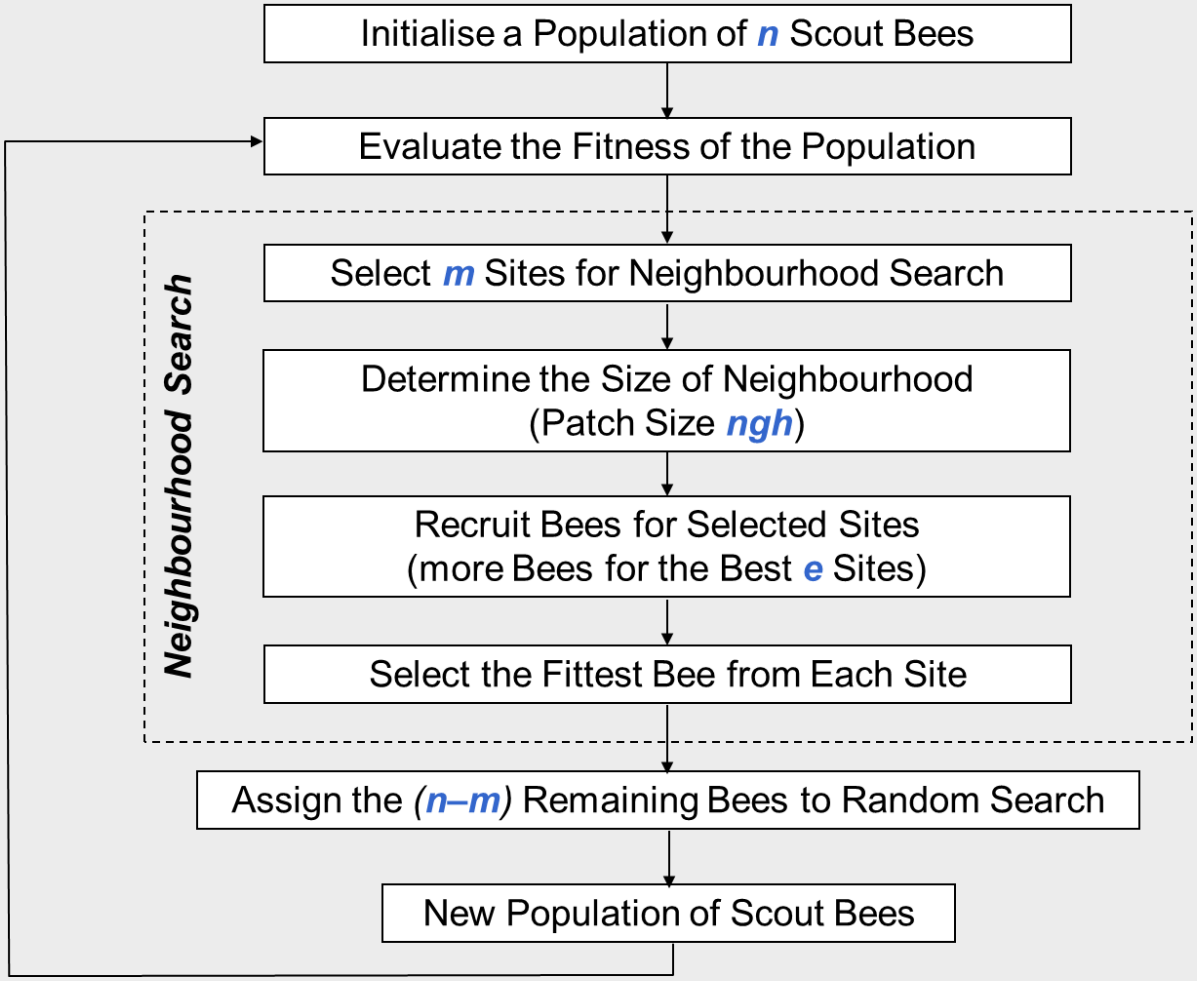
**1.** Initialise population with random solutions.
**2.** Evaluate fitness of the population.
**3.** While (stopping criterion not met)
    //Forming new population.
    **4.** Select sites for neighbourhood search.
    **5.** Recruit bees for selected sites (more bees for best **e** sites) and evaluate fitnesses.
    **6.** Select the fittest bee from each patch.
    **7.** Assign remaining bees to search randomly and evaluate their fitnesses.
**8.** End While.

# ABC Algorithm (Cont.)

The algorithm requires a number of parameters to be set:

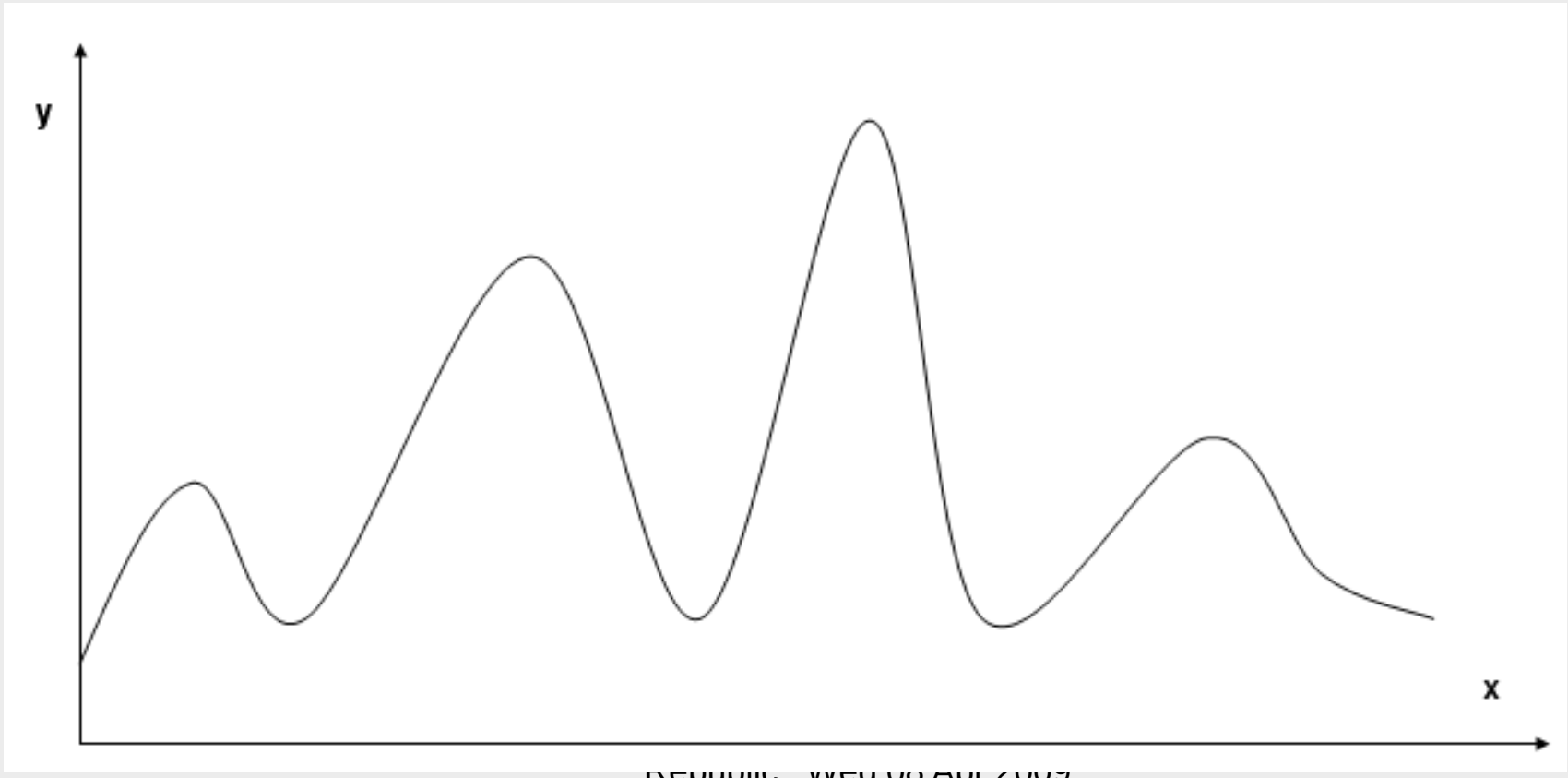
- *Number of scout bees **n*** → 100
- *Number of sites selected **m** out of **n** visited sites* → 10
- *Number of best sites **e** out of **m** selected sites* → 3
- *Number of bees recruited for best **e** sites **nep** or (**n2**)* Rich → 40 in neighborhood area
- *Number of bees recruited for the other (**m-e**) selected sites which is **nsp** or (**n1**)* Poor → 20
- *Initial size of patches **ngh** which includes site and its neighbourhood and stopping criterion* → 0-1 (0.2)
- *Number of algorithm steps repetitions **imax*** 10,300,1000

# ABC Algorithm: Flowchart

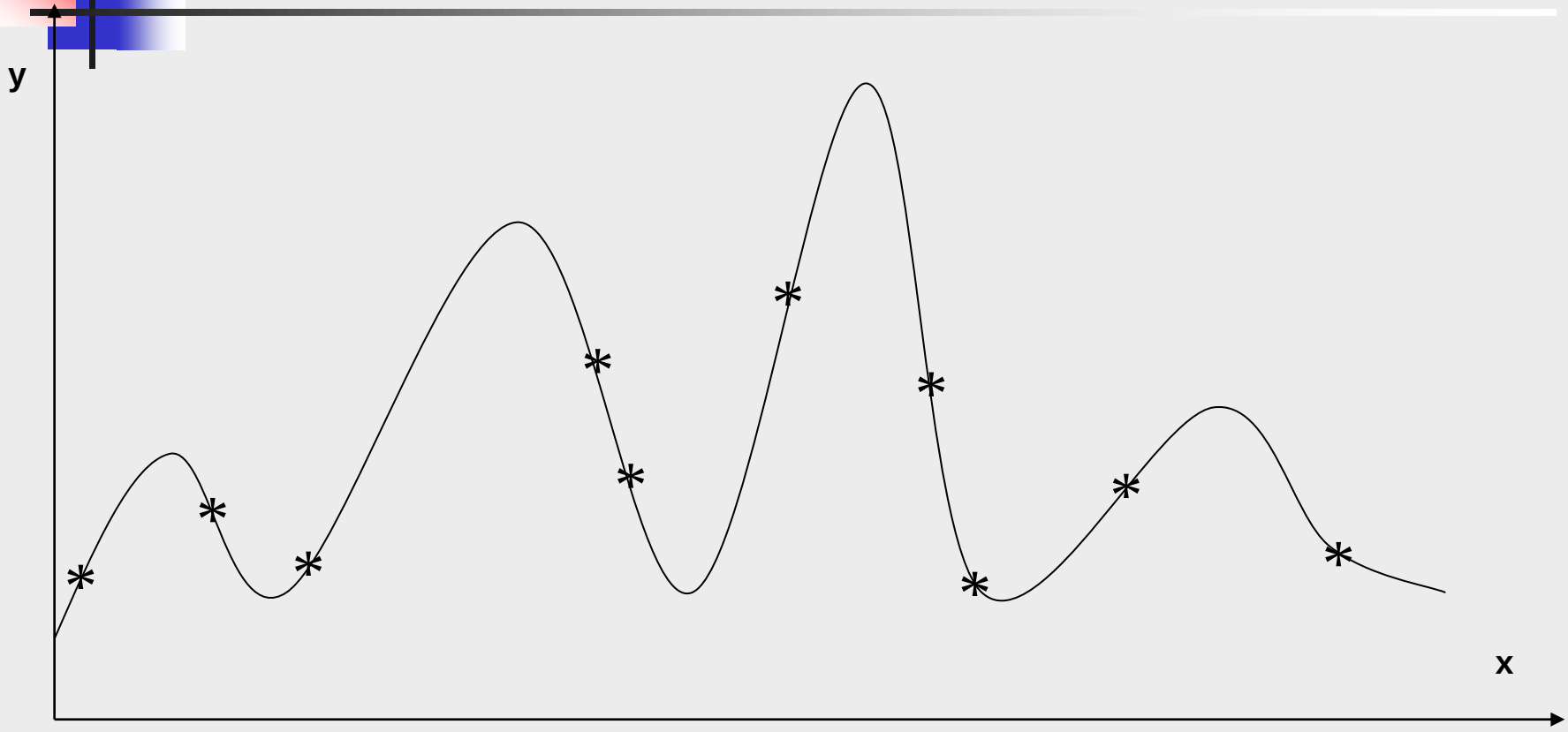


# Simple Example

- The following figure shows the mathematical function



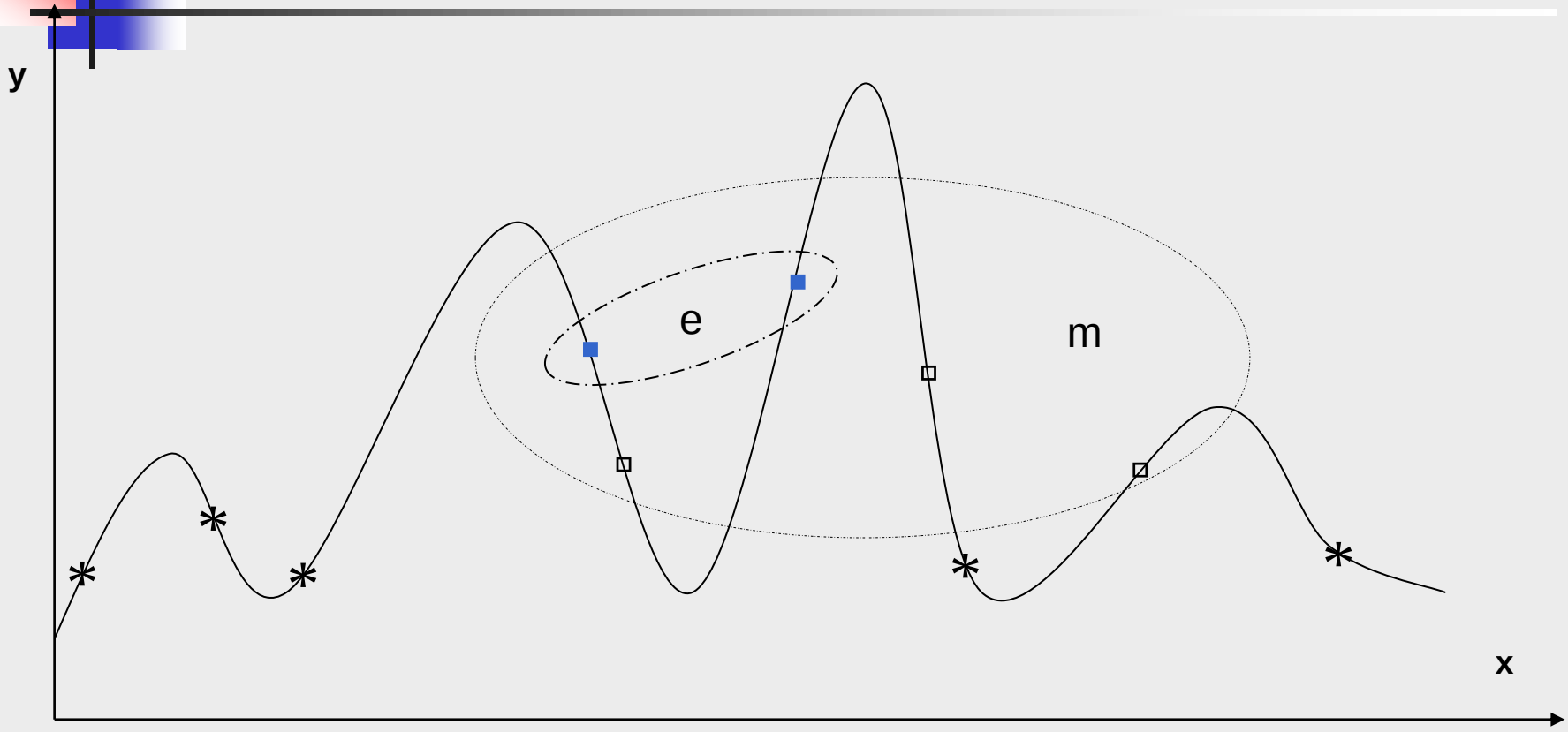
# Simple Example




Graph 1. Initialise a Population of (**n=10**) Scout Bees with random Search and evaluate the fitness.

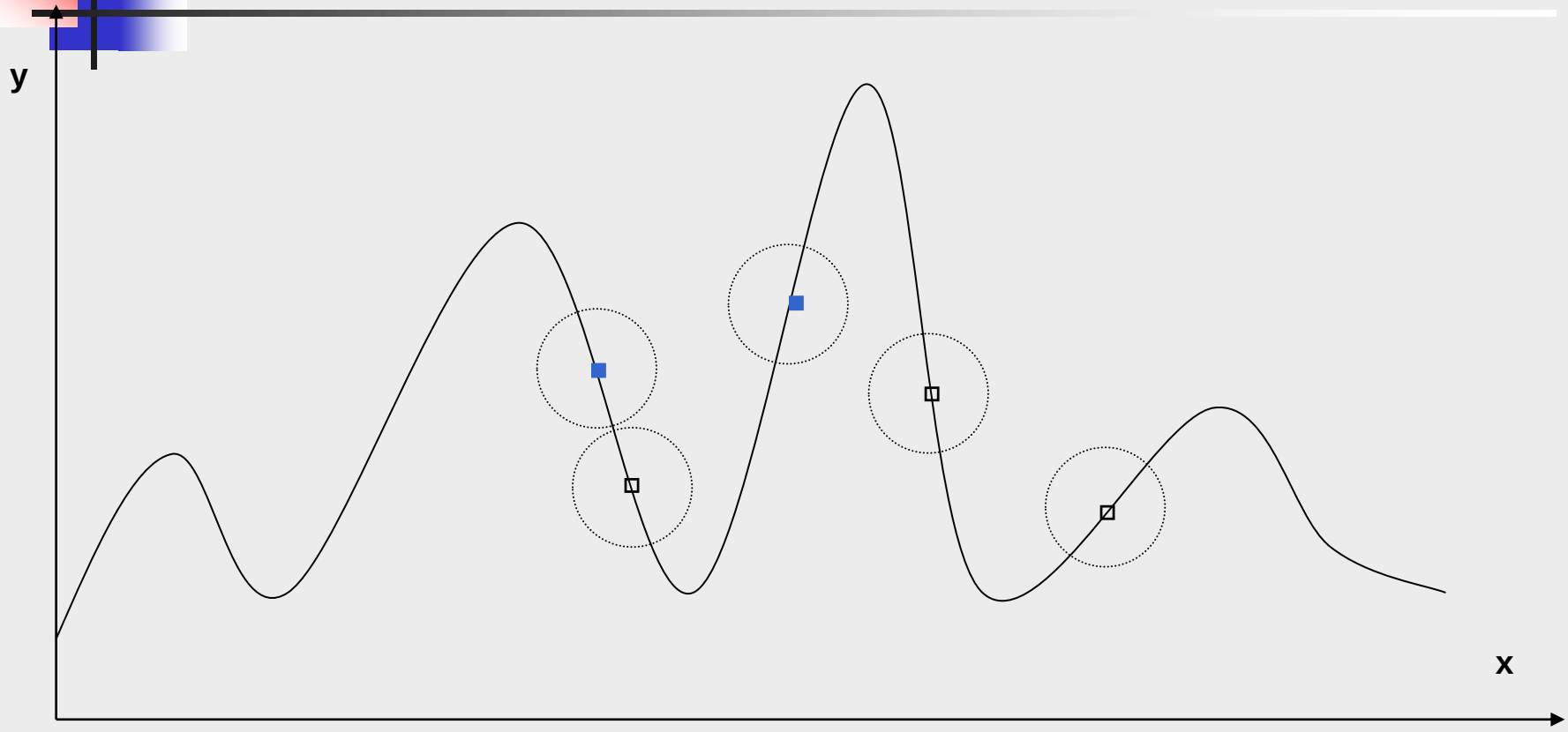
Masaryk University, Brno, Czech Republic , Wed 08 Apr 2009

# Simple Example



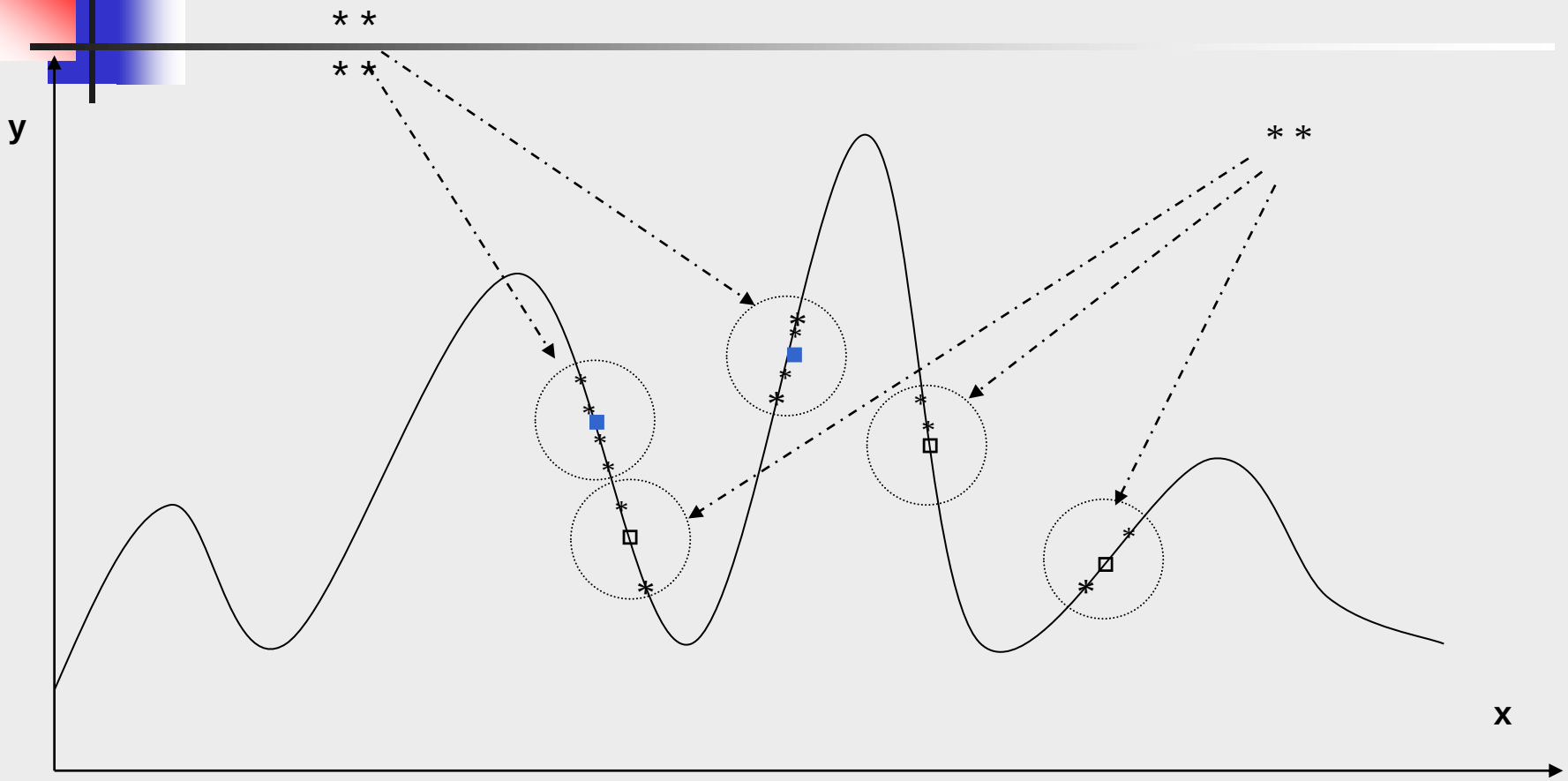

Graph 2. Select best (**m=5**) Sites for Neighbourhood Search:
(**e=2**) elite bees "▪" and (**m-e=3**) other selected bees"▫"

# Simple Example



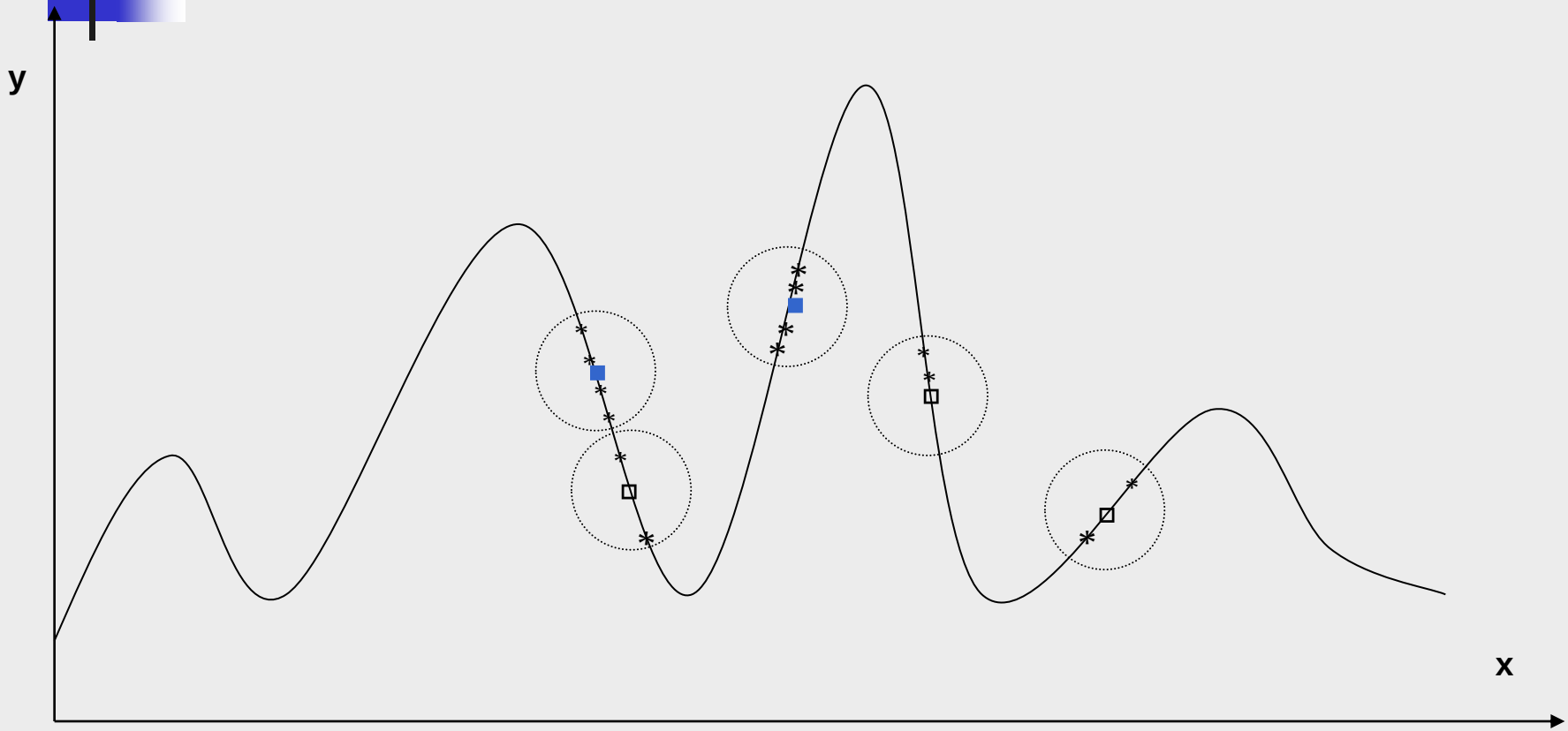

Graph 3. Determine the Size of Neighbourhood (Patch Size **ngh**)

# Simple Example



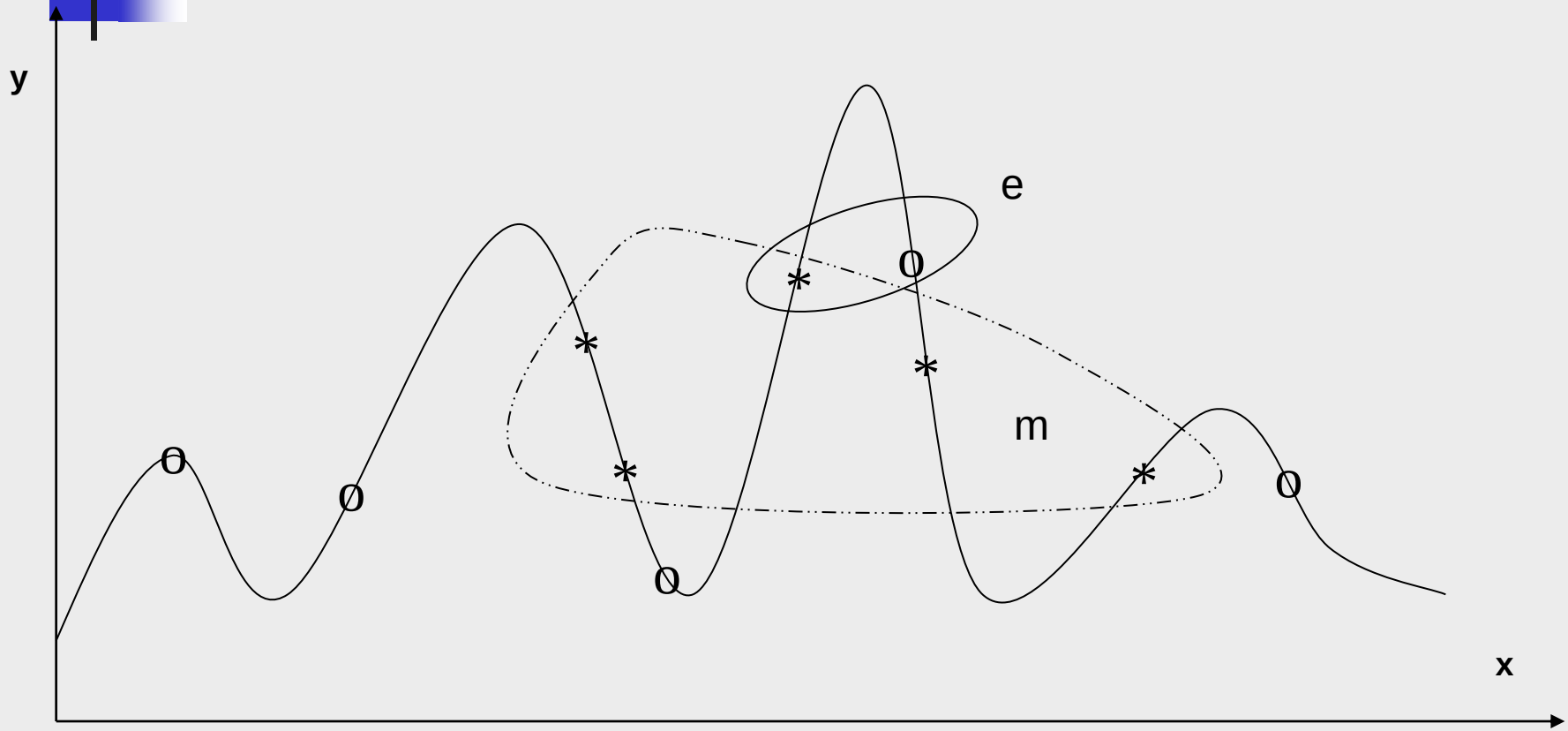

Graph 4. Recruit Bees for Selected Sites
(more Bees for the e=2 Elite Sites)

# Simple Example



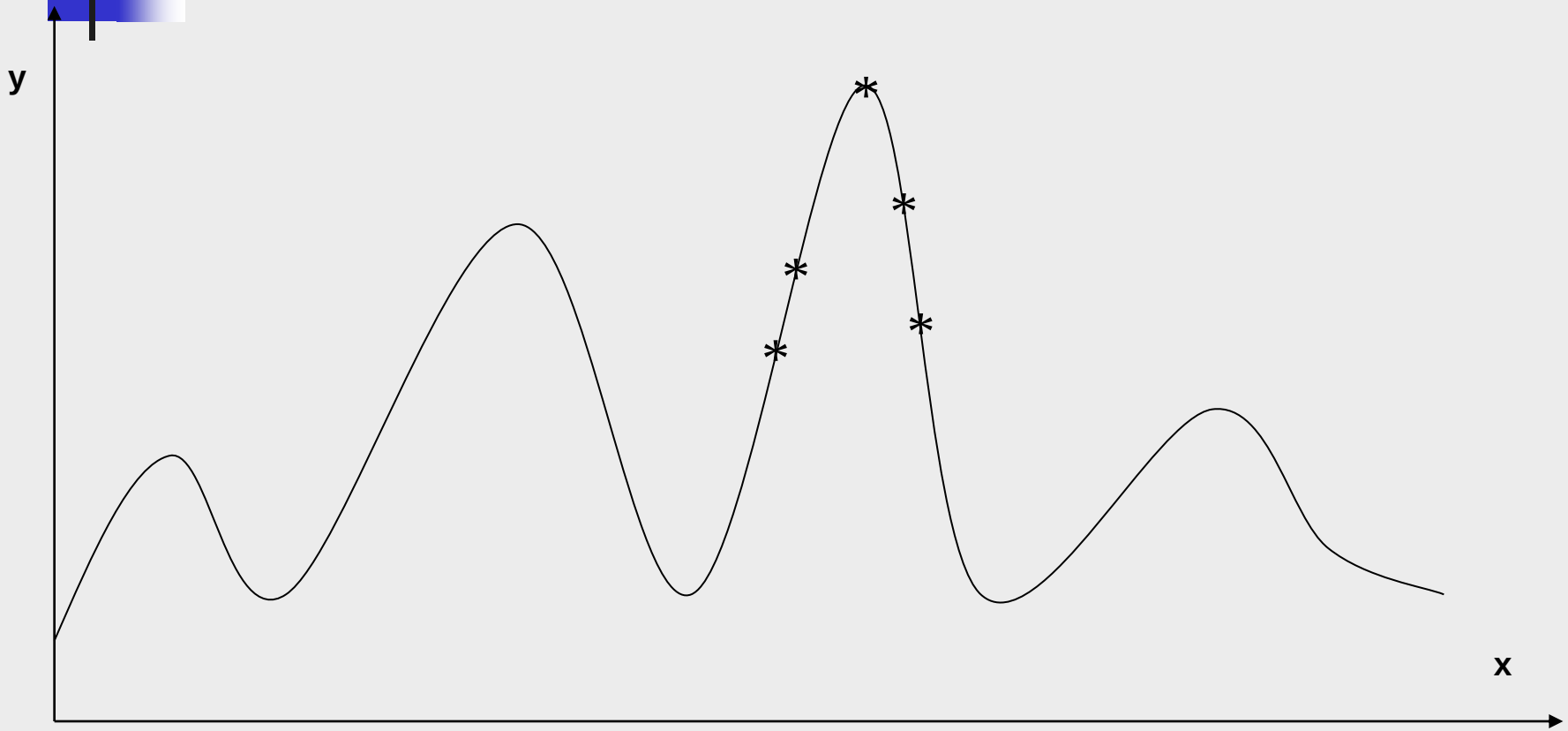

Graph 5. Select the Fittest Bee * from Each Site

# Simple Example



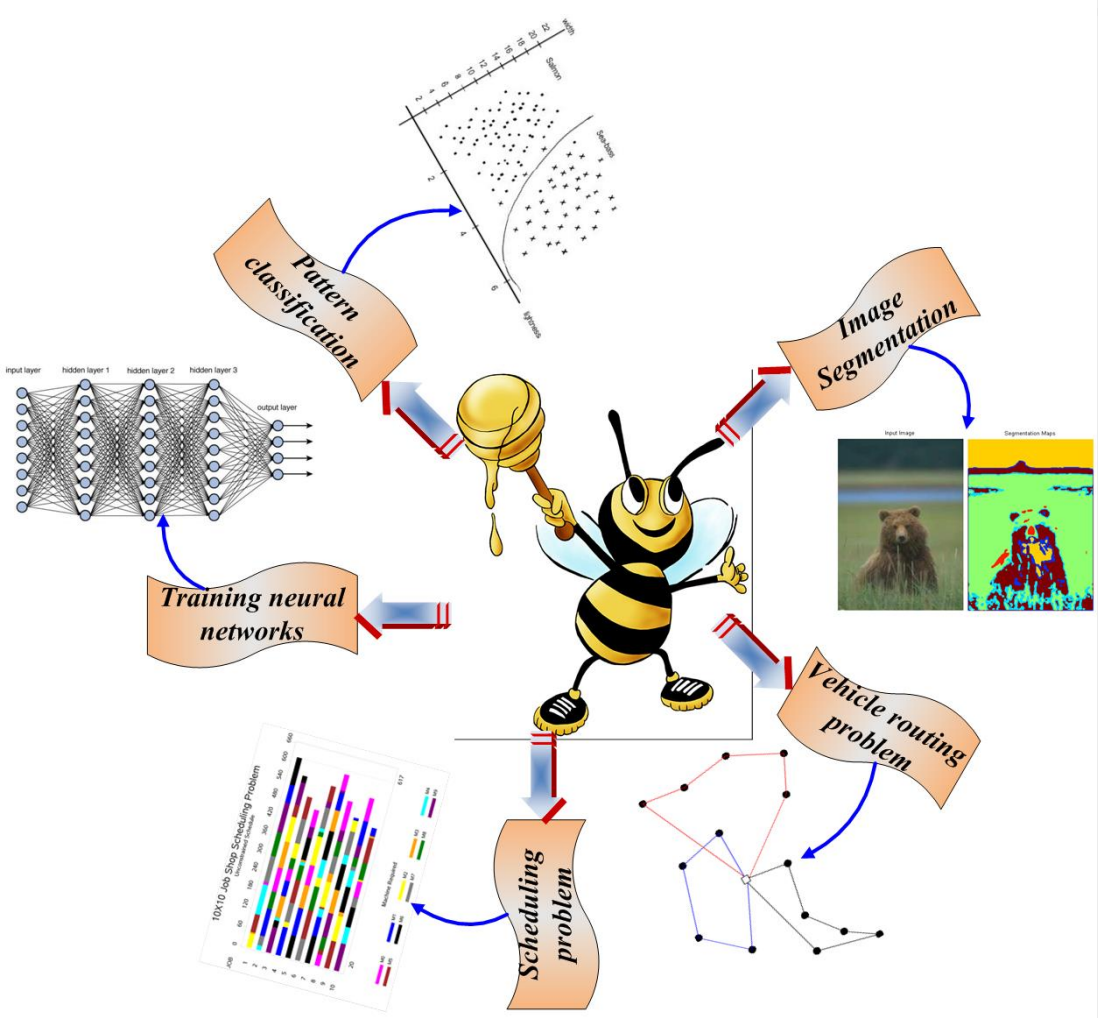

Graph 6. Assign the (**n–m**) Remaining Bees to Random Search

# Simple Example




Graph 7. Find The Global Best point

# Applications of ABC Algorithm




Many More Applications

https://transpireonline.blog/2019/08/02/artificial-bee-colony-abc-algorithm-a-novel-method-motivated-from-the-behavior-of-bees-for-optimal-solution/


## Open Discussion