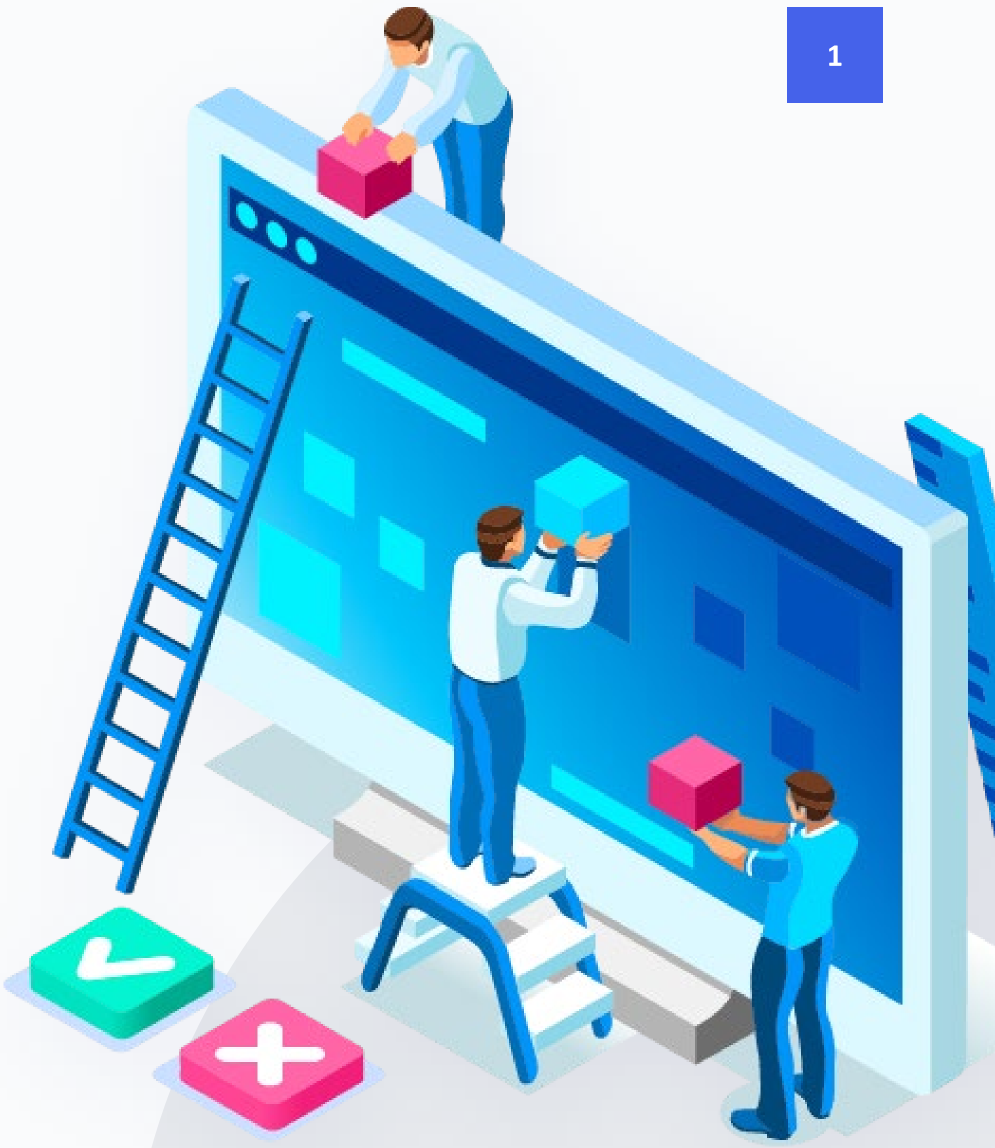
IE5600 Applied Programming | 2020

Meta-heuristics II

Che Yuxin



Agenda Today

- Che Yuxin will discuss about population-based metaheuristics
- Pan Binbin will discuss about the challenges of the projects

Course Logistics - Update

11:30/10

Meta-heuristics II – Population Based Heuristics

Group Project 3 – Time Dependent VRP using two of the above methods

Deadline 06/11 2359hrs

12:06/11

Class Presentation of Project 2 and Group discussion about the projects

Group Project 4 – Comparing the previous methods that have been implemented

Deadline 20/11 2359hrs

Group Project 5 – Project Summary Report

Deadline 20/11 2359hrs

13:13/11

Programming Test (Individual)

3-hour programming test – Covering Greedy, Dynamic Programming, Search, Graph and Optimization

01

Genetic Algorithm (GA)

02

Particle Swarm Intelligence (PSO)

03

Ant Colony Optimization (ACO)

1 Genetic Algorithm

Genetic Algorithm

Genetic Algorithms (GA) are search based algorithms based on the concepts of natural selection and genetics.

GAs are a subset of a much larger branch of computation known as Evolutionary Computation.

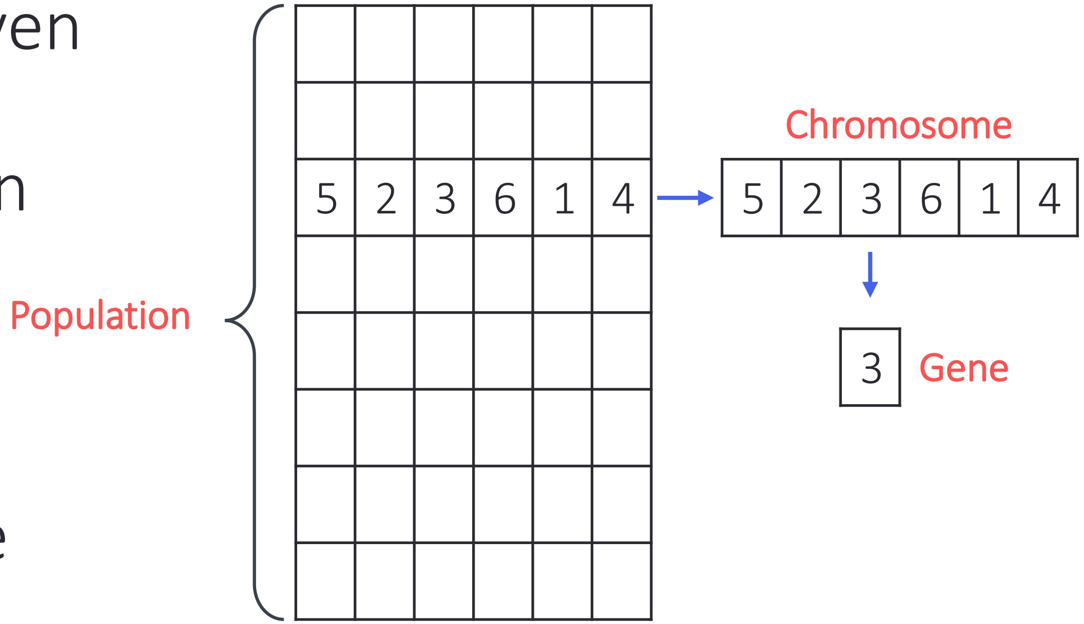
Genetic Algorithm

Darwinian Theory of "Survival of the Fittest".

- A population of possible solutions to the given problem are constructed at first. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations.
- Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals.
- In this way, we keep "evolving" better individuals or solutions over generations, till we reach a stopping criterion.

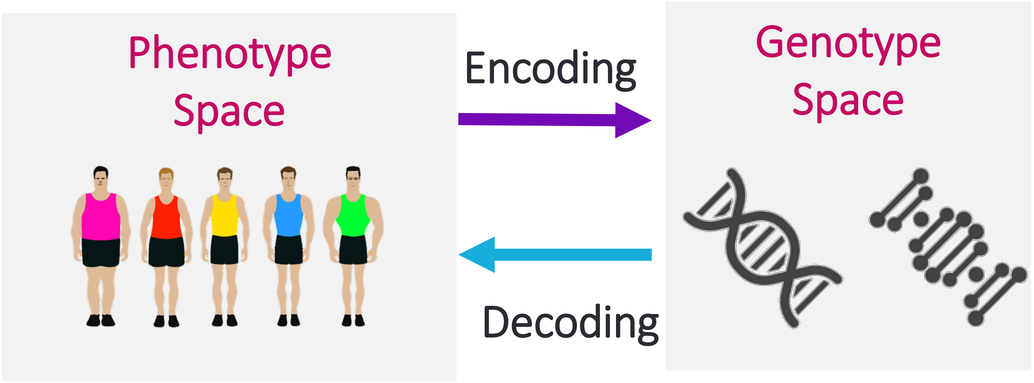
Genetic Algorithm

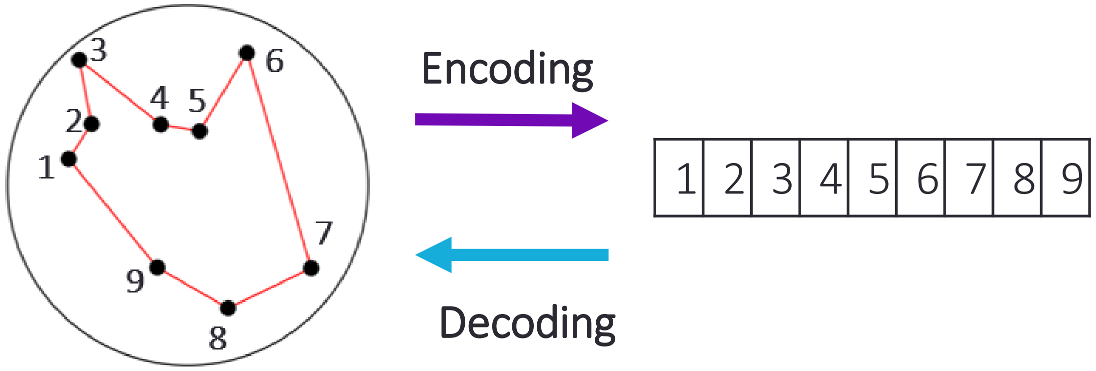
- Population – It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- Chromosomes – A chromosome is one such solution to the given problem.
- Gene – A gene is one element position of a chromosome.



Genetic Algorithm

- Genotype – Genotype is the population in the computation space.
- Phenotype – Phenotype is the population in the actual real world solution.
- Decoding and Encoding -- Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. (Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.)



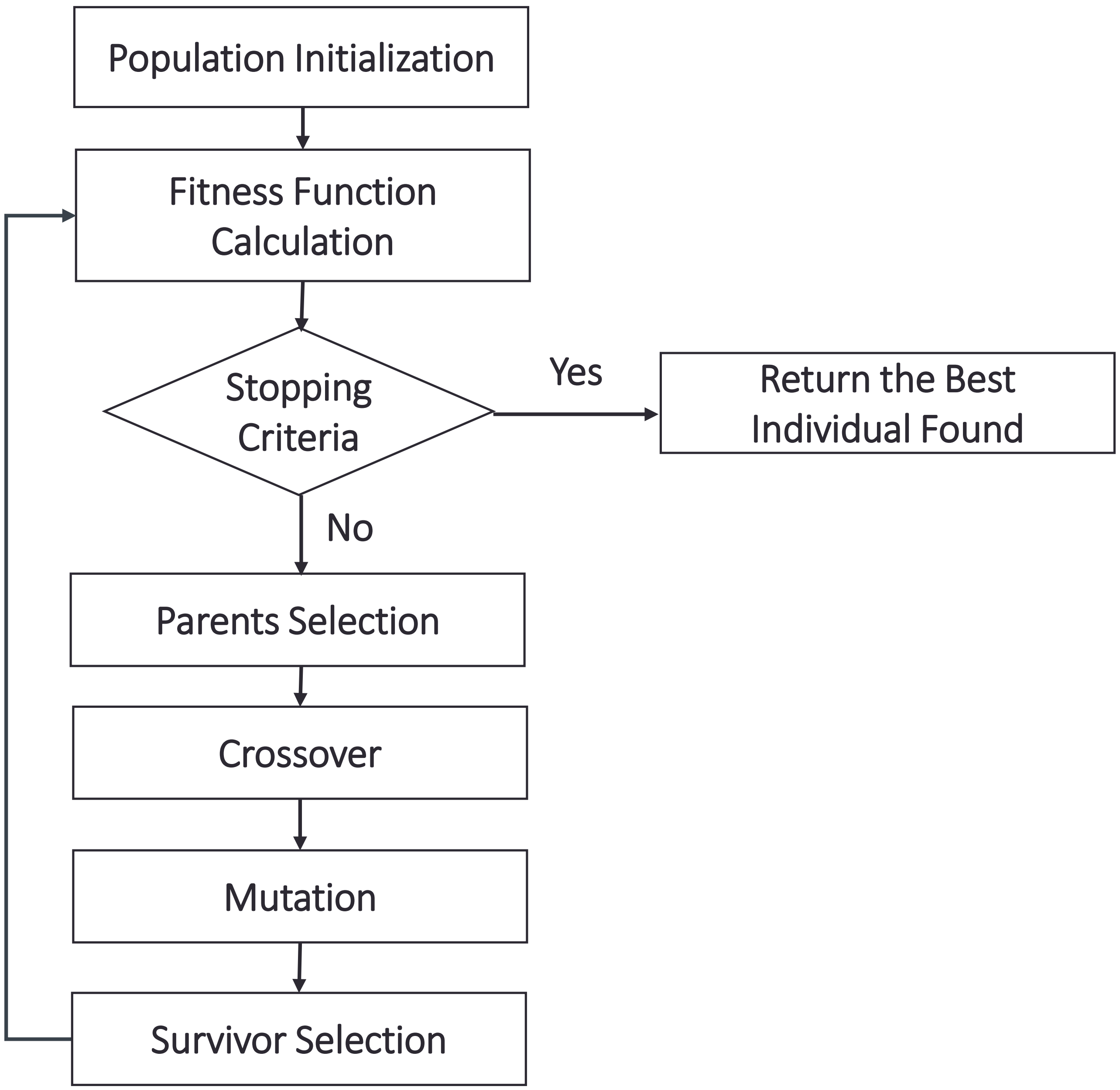


Genetic Algorithm

- Fitness Function – A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output.
- Genetic Operators – These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

gene-editing technology

Genetic Algorithm



Genetic Algorithm --- Genotype Representation

1. Binary Representation

For some problems when the solution space consists of Boolean decision variables – yes or no, the binary representation is natural.

- Take the 0/1 Knapsack Problem for example . If there are n items, we can represent a solution by a binary string of n elements, where the x_th element tells whether the item x is picked (1) or not (0).

0	1	1	0	1	1	1	0	1	0

Genetic Algorithm --- Genotype Representation

2. Real Valued Representation

For problems where we want to define the genes using continuous rather than discrete variables, the real valued representation is the most natural.

$$Min\ \ f(s) = \sum a_i s_i$$

0.2	0.5	0.6	0.1	0.3	0.4	0.8	0.2	0.5	0.9

Genetic Algorithm --- Genotype Representation

3. Integer Representation

For discrete valued genes, we cannot always limit the solution space to binary ‘yes’ or ‘no’, integer representation is desirable..

- For example, if we want to encode the four directions – North, South, East and West, we can encode them as {0,1,2,3}.

1	2	0	3	1	2	0	3	0	2

Genetic Algorithm --- Genotype Representation

4. Permutation Representation

For problems where the solution is represented by an order of elements, permutation representation is the most suited.

Take TSP for example, the solution is naturally an ordering or permutation of all the cities.

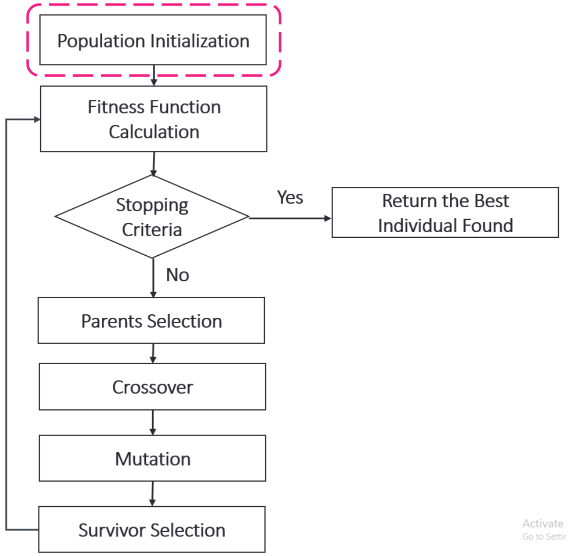
2	3	5	4	1	9	7	8	6	0

Genetic Algorithm -- Population Initialization

- Random Initialization
- Heuristic initialization – (If entire population should not be initialized using a heuristic, as it can result in the population having similar solutions and very little diversity)

Note:

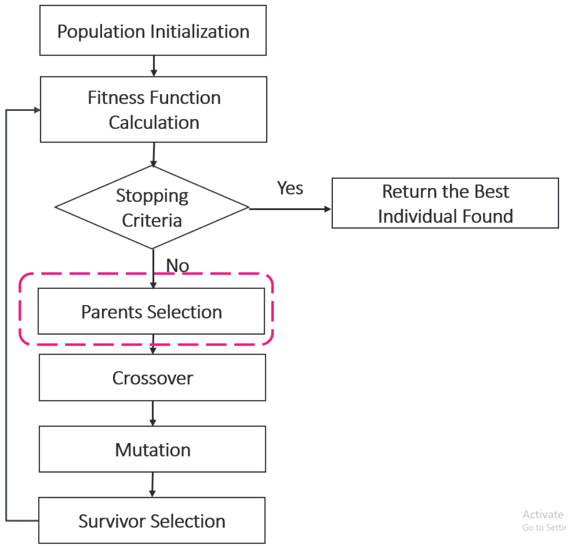
- The diversity of the population
- The population size

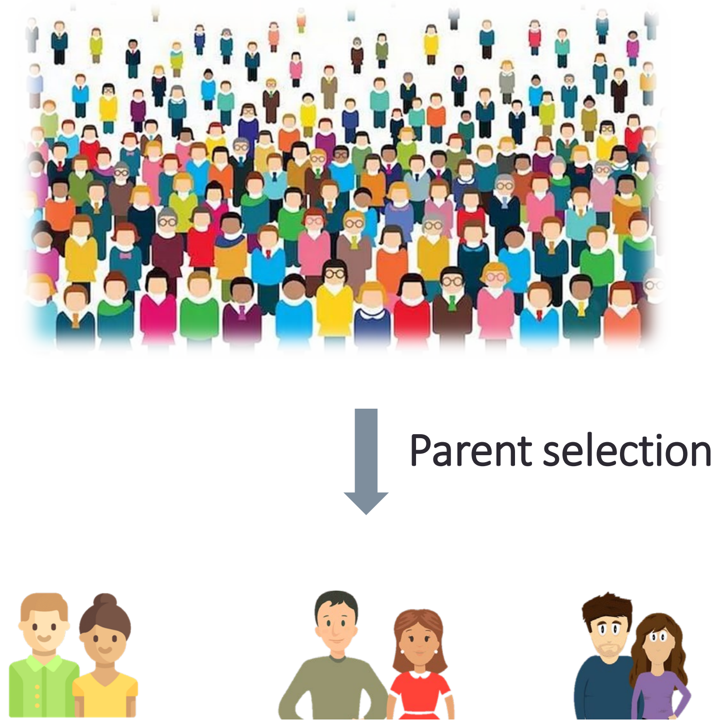


Genetic Algorithm -- Parent Selection

Parent Selection is the process of selecting parents which mate and recombine to create off-springs for the next generation.

- Diversity VS. Fitness





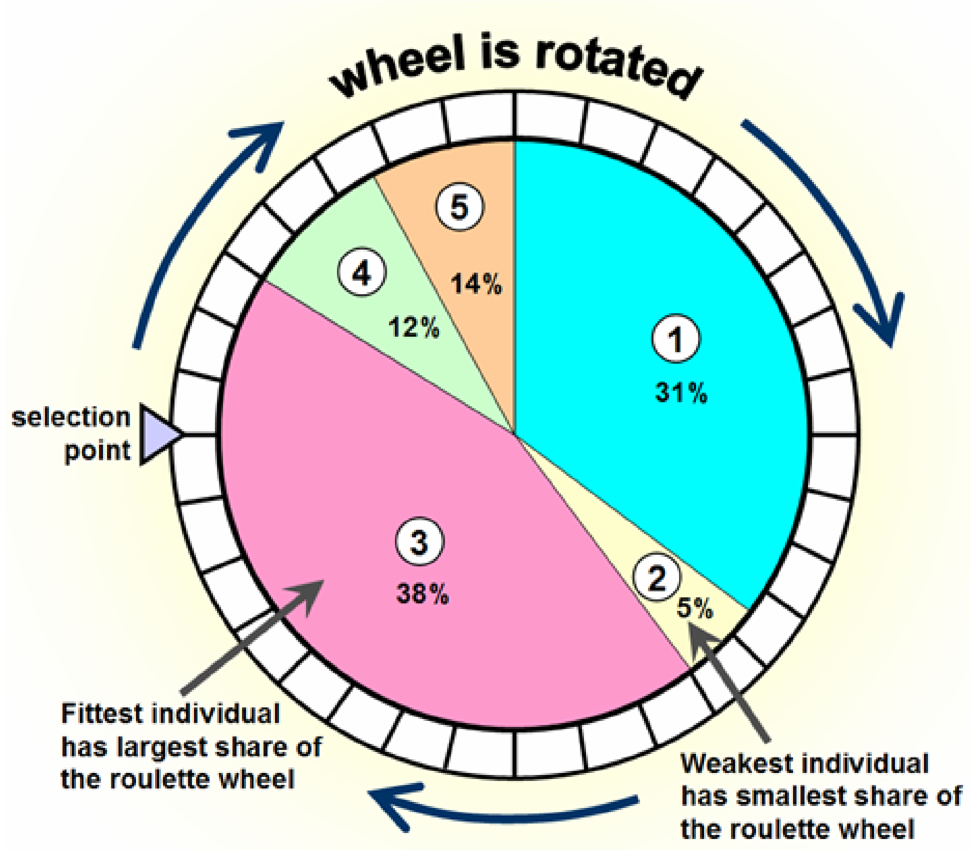
Genetic Algorithm -- Parent Selection

1. Fitness Proportionate Selection

Fitter individuals have a higher chance of mating and propagating their features to the next generation.

Roulette Wheel Selection

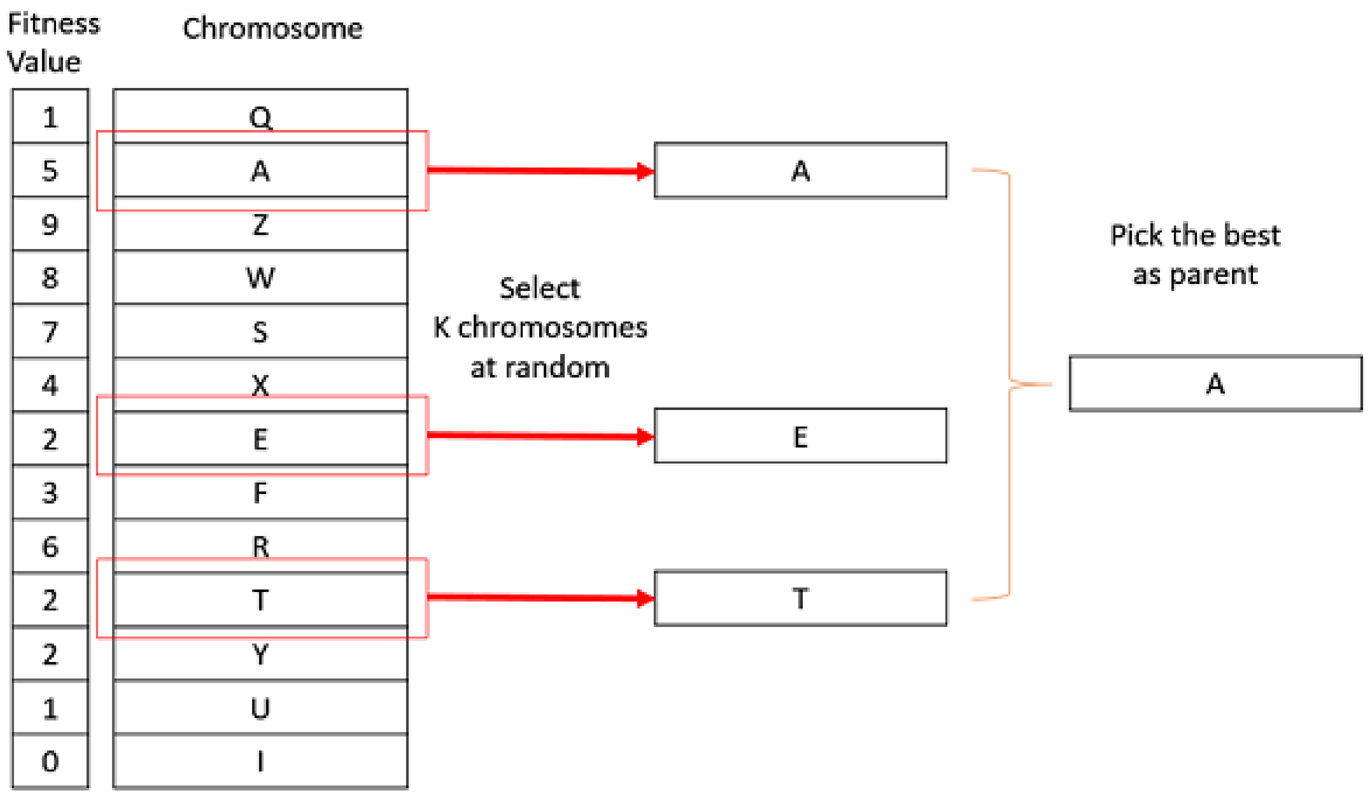
Consider a circular wheel. The wheel is divided into n pies, where n is the number of individuals in the population. Each individual gets a portion of the circle which is proportional to its fitness value.



Genetic Algorithm -- Parent Selection

2. Tournament Selection

In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent.



Genetic Algorithm -- Parent Selection

3. Rank Selection

Every individual in the population is ranked according to their fitness. The selection of the parents depends on the rank of each individual and not the fitness. The higher ranked individuals are preferred more than the lower ranked ones.

New fitness

Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

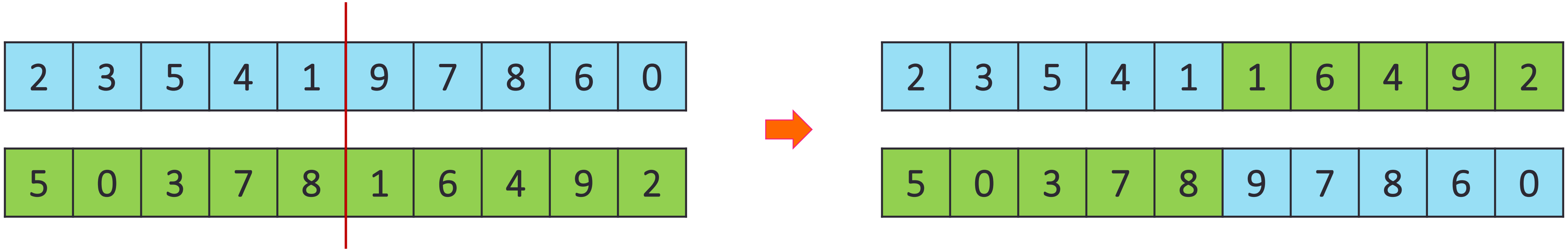
4. Random Selection

In this strategy we randomly select parents from the existing population.

Genetic Algorithm -- Crossover

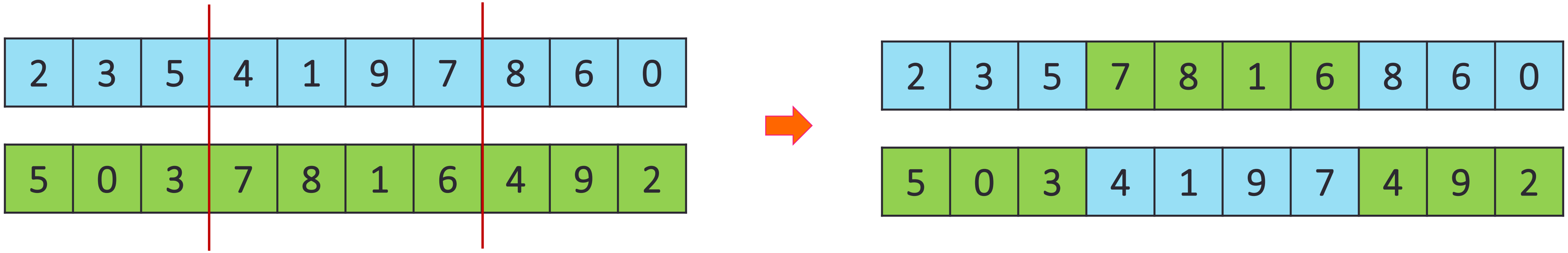
1. One Point Crossover

In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.



2. Multi Point Crossover

Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



Genetic Algorithm -- Crossover

3. Uniform Crossover

Don’t divide the chromosome into segments, rather treat each gene separately.

2	3	5	4	1	9	7	8	6	0
5	0	3	7	8	1	6	4	9	2

→

2	0	5	4	8	1	7	4	6	2
5	3	3	7	1	9	6	8	9	0

4. Whole Arithmetic Recombination

Take the weighted average of the two parents by using the following formula

$$Child1 = \alpha x + (1-\alpha)y$$
$$Child2 = \alpha y + (1-\alpha)x$$

0.2	0.5	0.6	0.1	0.3	0.4	0.8	0.2	0.5	0.9
0.5	0.3	0.6	0.9	0.1	0.6	0.4	0.7	0.8	0.3

α =0.2 →

0.44	0.34	0.6	0.74	0.14	0.56	0.48	0.6	0.74	0.42
0.26	0.46	0.6	0.26	0.26	0.44	0.72	0.3	0.56	0.78

Genetic Algorithm -- Crossover

5. Davis' Order Crossover (OX1)

OX1 is used for permutation based crossovers with the intention of transmitting information about relative ordering to the off-springs.

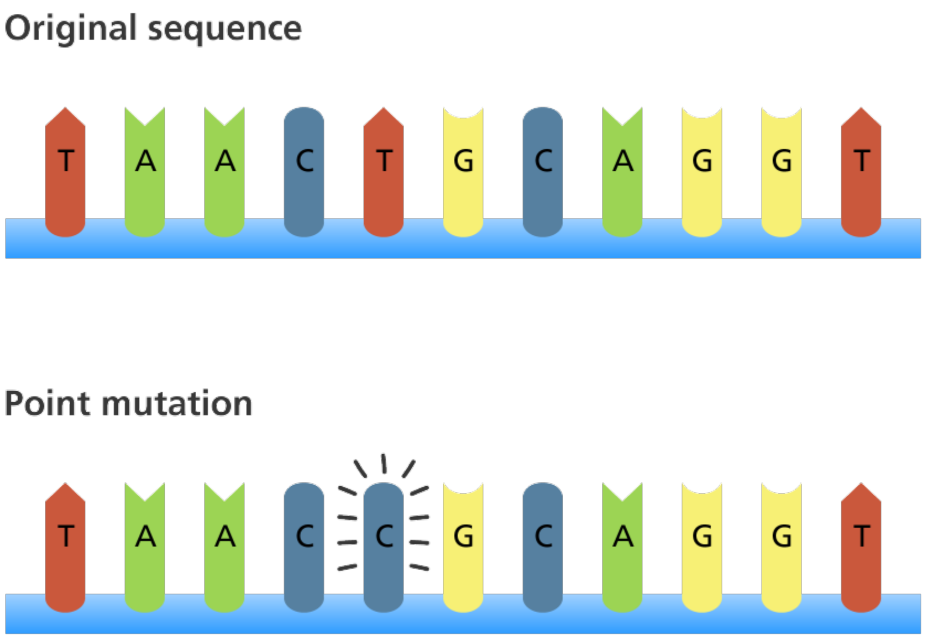
1. Create two random crossover points in the parent and copy the segment between them from the first parent to the first offspring.
2. Now, starting from the second crossover point in the second parent, copy the remaining unused numbers from the second parent to the first child, wrapping around the list.
3. Repeat for the second child with the parent's role reversed.

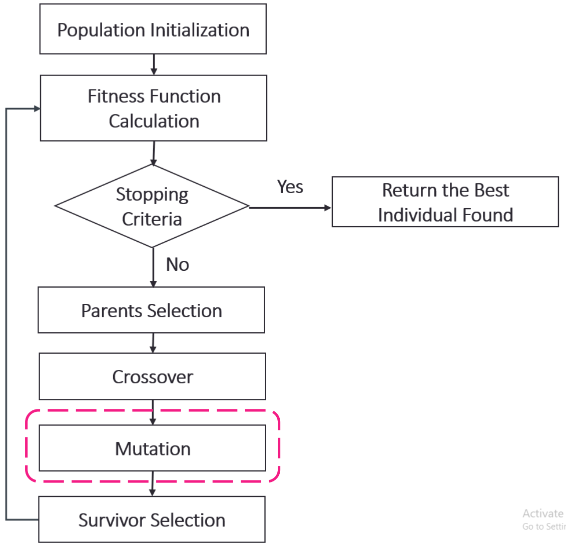
2	3	5	4	1	9	7	8	6	0
5	0	3	7	8	1	6	4	9	2

3	8	6	4	1	9	7	2	5	0
5	4	9	7	8	1	6	0	2	3

Genetic Algorithm -- Mutation

Mutation may be defined as a small random tweak in the chromosome, to get a new solution. It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability.





Genetic Algorithm -- Mutation

1. Bit Flip Mutation

Select one or more random bits and flip them. This is used for binary encoded GAs.

0	1	1	0	1	1	1	0	1	0

→

0	1	1	0	0	1	1	0	1	0

2. Random Resetting

A random value from the set of permissible values is assigned to a randomly chosen gene. This is used for real valued representation.

Genetic Algorithm -- Mutation

3. Swap Mutation

Select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings.

2	3	5	4	1	9	7	8	6	0	→	2	3	7	4	1	9	5	8	6	0

4. Scramble Mutation

From the entire chromosome, a subset of genes is chosen, and their values are scrambled or shuffled randomly. This is common in permutation based encodings.

2	3	5	4	1	9	7	8	6	0	→	2	3	1	4	5	7	9	8	6	0

Genetic Algorithm -- Survivor Selection

5. Inversion Mutation

Select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.

2	3	5	4	1	9	7	8	6	0

→

2	3	7	9	1	4	5	8	6	0

Genetic Algorithm -- Survivor Selection

The Survivor Selection Policy determines which individuals are to be kicked out and which are to be kept in the next generation. It is crucial as it should ensure that the fitter individuals are not kicked out of the population, while at the same time diversity should be maintained in the population.

- **Elitism**. In simple terms, it means the current fittest member of the population is always propagated to the next generation. Therefore, under no circumstance can the fittest member of the current population be replaced.
- **Random selection**. The easiest policy is to kick random members out of the population, but such an approach frequently has convergence issues.

Genetic Algorithm -- Survivor Selection

Age Based Selection

It is based on the premise that each individual is allowed in the population for a finite generation where it is allowed to reproduce, after that, it is kicked out of the population no matter how good its fitness is.

Age	Fitness Value	Chromosome
6	1	P1
8	5	P2
5	9	P3
10	8	P4
2	7	P5
3	4	P6
9	2	P7
2	2	P8
5	1	P9
4	0	P10

Population

Fitness Value	Chromosome
5	C1
9	C2

Offspring

Age	Fitness Value	Chromosome
6	1	P1
8	5	P2
5	9	P3
0	5	C1
2	7	P5
3	4	P6
0	9	C2
2	2	P8
5	1	P9
4	0	P10

New Population

Genetic Algorithm -- Survivor Selection

Fitness Based Selection

In this fitness based selection, the children tend to replace the least fit individuals in the population.

Fitness Value	Chromosome
1	P1
5	P2
9	P3
8	P4
7	P5
4	P6
2	P7
2	P8
1	P9
0	P10

Population

Fitness Value	Chromosome
8	C1
9	C2

Offspring

Fitness Value	Chromosome
8	C1
5	P2
9	P3
8	P4
7	P5
4	P6
2	P7
2	P8
1	P9
9	C2

New Population

Genetic Algorithm

How to encode a solution of CVRP?

1. Encoding: A sequence of n nodes
   Decoding: split into several trips

Rank	1	2	3	4	5	6	7	8	9
P1	4	5	6	7	8	9	2	3	1
P2	3	5	6	8	9	7	4	1	2

2. Encoding: assignments to different vehicles
   Decoding: consider traveling nodes in the same vehicle as a TSP problem

Rank	1	2	3	4	5	6	7	8	9
P1	1	1	1	2	2	2	3	3	3
P2	1	2	3	3	2	2	1	3	1

[1] Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & operations research*, *31*(12), 1985-2002.

[2] Baker, Barrie M., and M. A. Ayechew. "A genetic algorithm for the vehicle routing problem." *Computers & Operations Research* 30.5 (2003): 787-800.

[3] Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, *60*(3), 611-624.

2

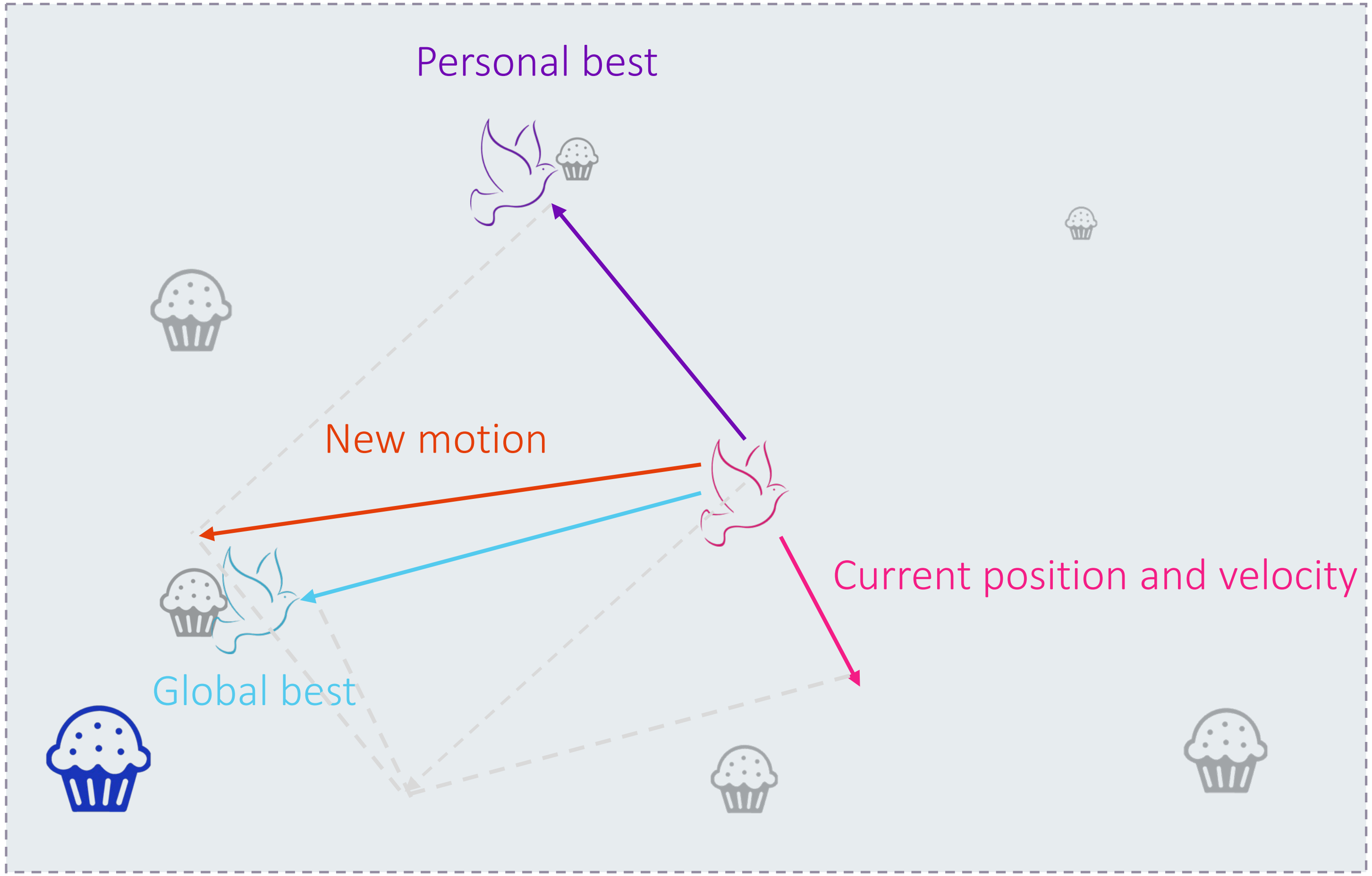
Particle Swarm Intelligence

Particle Swarm Intelligence

Particle Swarm uses the metaphor of the flocking behavior of birds to solve optimization problems.

In Particle Swarm Optimization(PSO) algorithm, many autonomous entities (particles)are stochastically generated in the search space. Each particle is a candidate solution to the problem, and is represented by a velocity, a location in the search space and has a memory which helps it in remembering its previous best position.

Particle Swarm Intelligence



- Objective: find the biggest cake
- Information:
  1) Current position and moving velocity
  2) Have memory of his personal best solution has been found
  3) Communicate by sound
  - Change direction every 1 minute

Particle Swarm Intelligence

- A swarm consists of N particles flying around in a D-dimensional search space.
- Every particle swarm has some sort of topology describing the interconnections among the particles.
- The set of particles to which a particle i is topologically connected is called its neighborhood. The neighborhood may be the entire population or some subset of it.
- The two most commonly used neighbors are known as g_{best} (for "global best") and l_{best} (for "local best").

In the initialization phase of PSO, the positions and velocities of all individuals are randomly initialized.

At each iteration t, a particle i adjusts its position $X_i(t)$ and velocity $V_i(t)$ along each dimension d of the search space, based on the best position $P_i(t)$ that it has encountered so far in its flight(also called the personal best for the particle) and the global best position $P_g(t)$ found by any other particle in its topological neighborhood.

Particle Swarm Intelligence

The velocity defines the direction and the distance the particle should go:

$$V_{\mathrm{i}}(t+1) = V_i(t) + C_1\varphi_1\big(P_i(t) - X_i(t)\big) + C_2\varphi_2(P_g(t) - X_i(t))$$

The position of each particle is also updated in each iteration by adding the velocity vector to the position vector:

$$X_i(t+1) = X_i(t) + V_i(t+1)$$

where $i = 1, 2, \ldots, N$, and N is the size of the swarm; φ_1 and φ_2 are two random numbers uniformly distributed in the range[0, 1], C_1 and C_2 are constant multiplier terms known as acceleration coefficients.

Particle Swarm Intelligence

PSO

Initialize a population of particles with random positions and velocities on D dimensions in the search space

while termination condition not met **do**

  for each particle i **do**

    Update velocity of the particle

    Update the position of the particle

    Evaluate the fitness $f(X_i)$

    if $f(X_i) < f(P_i)$ **then**

      $P_i \leftarrow X_i$ // update personal best

    end

    if $f(X_i) < f(P_g)$ **then**

      $P_g \leftarrow X_i$ // update global best

    end

  end

end

Particle Swarm Intelligence

How to use PSO to solve VRP?

$X_i \longleftrightarrow$ Current solution

$V_i \longleftrightarrow$ How to modify the current solution

[1] Chen, A. L., Yang, G. K., & Wu, Z. M. (2006). Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University-Science A*, *7*(4), 607-614.
[2] Ai, T. J., & Kachitvichyanukul, V. (2009). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering*, *56*(1), 380-387.

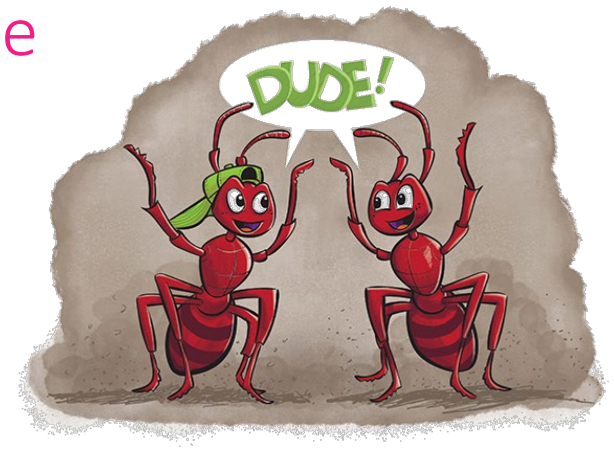
3

Ant Colony Optimization

Ant Colony Optimization

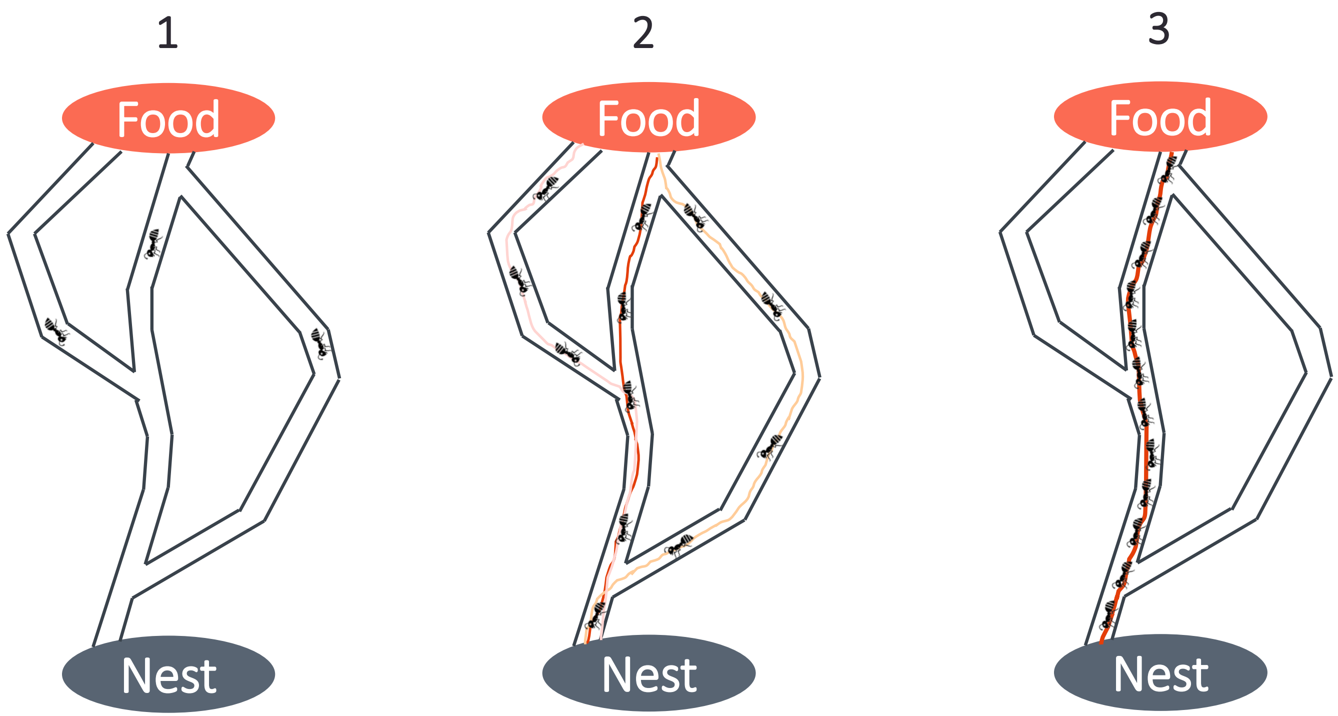
Ant Colony Optimization (ACO) is a genetic algorithm inspired by an ant’s natural behavior. To fully understand the ACO algorithm, we need to get familiar with its basic concepts:

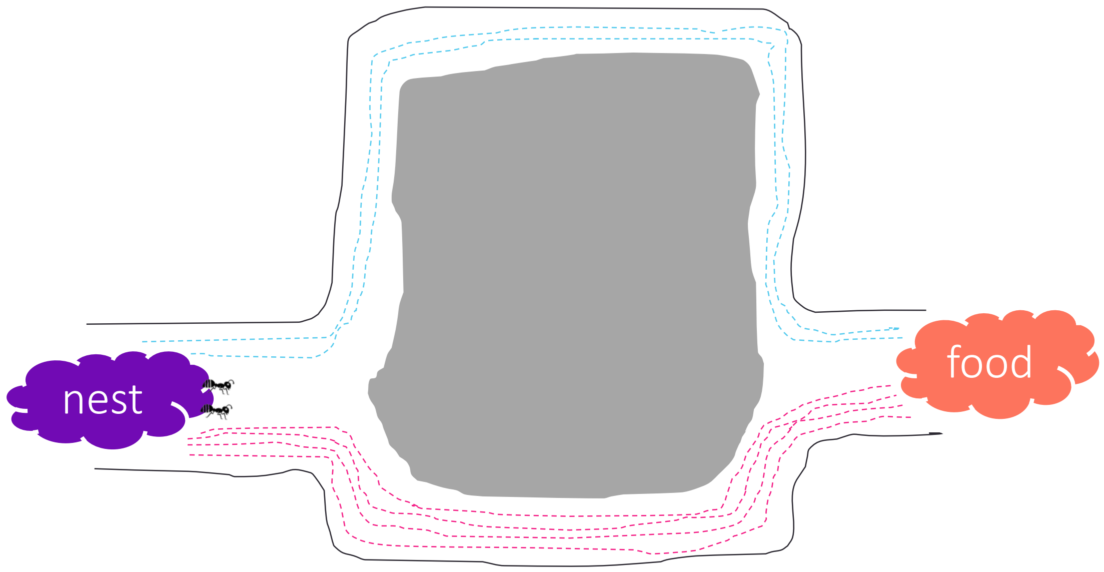
- ants use pheromones to find the shortest path between home and food source
- pheromones evaporate quickly
- ants prefer to use shorter paths with denser pheromone



Ant Colony Optimization

Ants live in colonies, and roam around their colonies to search for food. Once an ant find the food and return, it deposits pheromone on the paths based on the quantity and quality of the food. So, other ants can smell that and follow that path. The higher the pheromone level has a higher probability of choosing that path and the more ants follow the path, the amount of pheromone will also increase on that path.





Larger amount of pheromone, more ants will follow the path, which is the best path

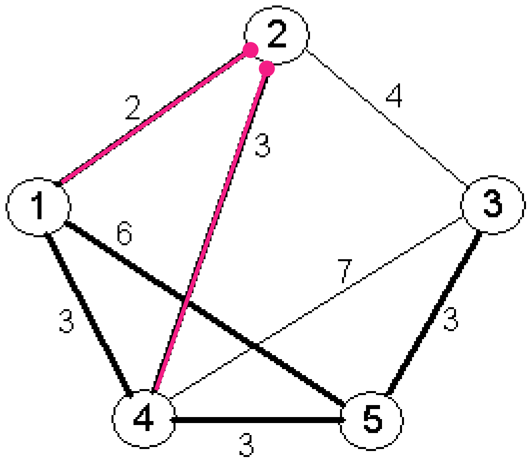
Ant Colony Optimization

ACO

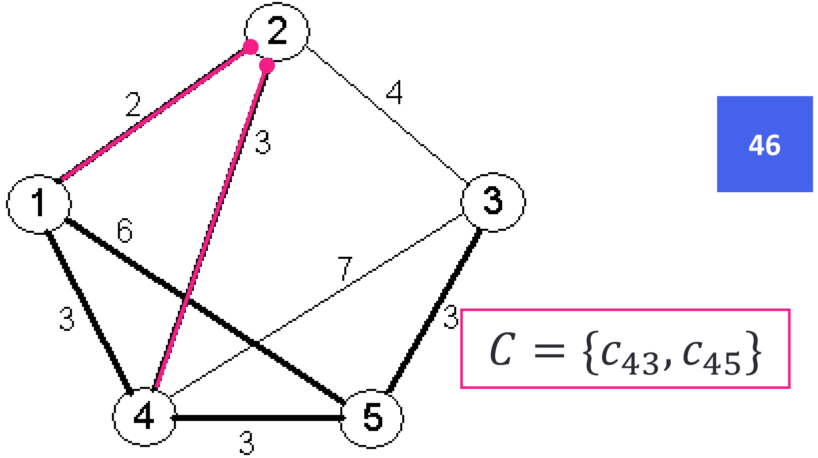
```
Initialize pheromone values
while termination condition not met do
    Construct Ants Solutions
    Daemon Actions
    Update Pheromones
end
```

We denote a solution component by c_i^j as the instantiation of a variable x_i with a particular value v_i^j. A pheromone value τ_{ij} is associated with each solution component c_i^j , this value serves as a form of memory, adapted over time to indicate the desirability of choosing solution component c_i^j.

Take TSP as an example, a component of the solution is a city that is added to a tour. Ants then need to appropriately combine solution components to form feasible walks.



Ant Colony Optimization



Construct Ant Solutions

- A set of m artificial ants incrementally and stochastically builds solutions to the considered problem starting from an initially empty partial solution $s_p = \emptyset$.
- At each construction step, the current partial solution s_p is extended by adding a feasible solution component c_i^j from the set $\mathcal{N}(s_p) \subseteq C$. ($c_i^j$ is consider to be the edge from node i to j)
- C denotes the set of all possible solution components and $\mathcal{N}(s_p)$ is defined as the set of components that can be added to the current partial solution s_p while maintaining feasibility.

Ant Colony Optimization

- Construct Ant Solutions (continued)

In order to choose, which of the available solution components c_i^j should be added to the current partial solution s_p, a probabilistic choice is made.

$$p\left(c_i^j \middle| s_p\right) = \frac{\tau_{ij}^{\alpha} \cdot \left[\eta\left(c_i^j\right)\right]^{\beta}}{\sum_{c_i^l \in \mathcal{N}(s_p)} \tau_{ij}^{\alpha} \cdot \left[\eta\left(c_i^l\right)\right]^{\beta}}, \forall c_i^j \in \mathcal{N}(s_p)$$

- τ_{ij} is the amount of pheromone on edge i, j
- α is a parameter to control the influence of τ_{ij}
- $\eta\left(c_i^j\right)$is the desirability of edge i, j (typically 1/ d_{ij})
- β is a parameter to control the influence of $\eta\left(c_i^j\right)$

Ant Colony Optimization

- Daemon Action

Once solutions have been constructed, and before updating the pheromone values, often some problem specific actions may be required. These are often called *daemon actions*, and can be used to implement problem specific and/or centralized actions, which cannot be performed by single ants. The most used daemon action consists in the application of **local search** to the constructed solutions: the locally optimized solutions are then used to decide which pheromone values to update.

- Update Pheromones

The aim of the pheromone update is to increase the pheromone values associated with good solutions, and to decrease those that are associated with bad ones.

1. decreasing all the pheromone values through pheromone evaporation (avoid a premature convergence of the algorithm to suboptimal solutions and then favoring the exploration of not yet visited areas of the search space.)
2. increasing the pheromone levels associated with a chosen set of good solutions S_{upd} through pheromone deposit (make these solution components more attractive for ants in the following iterations)

- Update Pheromones **(continued)**

The pheromone update is commonly implemented as

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s)$$

where S_{upd} is the set of good solutions that are used to deposit pheromone, $g(\cdot)$: $S \rightarrow R^+$ is a function such that $f(s) < f(s') \Rightarrow g(s) \geq g(s')$ is commonly called the quality function , $0 \leq \rho \leq 1$ is the pheromone evaporation rate.

- Update Pheromones **(simplified)**

The pheromone update is commonly implemented as

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum \nabla\tau_{ij}$$

- τ_{ij} is the amount of pheromone on a given edge i, j
- P is the rate of pheromone evaporation
- $\nabla\tau_{ij}$ is the amount of pheromone deposited, typically given by

$$\nabla\tau_{ij} = \begin{cases} \dfrac{1}{L_k} & if\ ant\ k\ travels\ on\ the\ edge\ i,j \\ 0 & otherwise \end{cases}$$

,where L_k is the cost of the k th ant's tour

Ant Colony Optimization

[1] ACO solving TSP. https://bsantosa.files.wordpress.com/2015/03/aco-tutorial-english2.pdf

[2] Mazzeo, S., & Loiseau, I. (2004). An ant colony algorithm for the capacitated vehicle routing. *Electronic Notes in Discrete Mathematics*, *18*, 181-186.

[3] Lee, Chou-Yuan, et al. "An enhanced ant colony optimization (EACO) applied to capacitated vehicle routing problem." *Applied Intelligence* 32.1 (2010): 88-95.

Classification for meta-heuristics

Local search	Global search
SA	ACO
Tabu	PSO
ILS	GA
VNS	
GRASP	

Single-solution	Population-based
SA	ACO
ILS	PSO
VNS	GA
GLS	

Please think about the similarities and differences among these meta-heuristic algorithms.

Reference

1. *Gendreau, M., & Potvin, J. Y. (Eds.). (2010). Handbook of metaheuristics (Vol. 2, p. 9). New York: Springer.*
2. *Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. Information Sciences, 237, 82–117. https://doi.org/10.1016/j.ins.2013.02.041*

Thank you!

Che Yuxin