Metaheuristic Techniques

GROUP B4

Introduction

Heuristic is defined as a method of denoting a rule of thumb for solving a problem without the exhaustive application of a procedure.

A heuristic method is one that

1. Looks for an approximate solution
2. Does not explore each and every possible solution in the search space before arriving at a solution.
3. Need not particularly have a mathematical convergence proof.

Introduction

Metaheuristic methods are particularly relevant in the context of solving search and optimization problems.

They describe methods that use one or more heuristics and therefore inherits properties mentioned above.

A metaheuristic methods:
1. Seeks to find a near optimal solution instead of the exact solution

2. Usually has no rigorous proof of convergence to the optimal solution.

3. Faster than exhaustive search.

Introduction

They are iterative in nature and use stochastic (having random probability distribution) methods in their search process to modify one or more candidate solutions.

They are popular due to the fact that they can handle different complexities associated with practical problems and arrive at a reasonably acceptable solution. They are able to solve non-convex NP hard problems.

Introduction

Advantages over classical optimization methods

1. Metaheuristics can lead to good enough solutions for computational for computational ease.
2. Metaheuristics can lead to good enough solutions for NP problems
3. They require no gradient information and therefore used with non-analytic, black-box or simulation based objective functions.
4. Have the ability to recover from local optima due to inherent stochasticity.
5. They can better handle uncertainties in objectives due to the ability to recover from local optima.
6. They can handle multiple objectives with few algorithmic changes.

Classification

Metaheuristic techniques can be classified based on:

1. Number of initial solutions modified
    a. Single solution - one solution that gets modified in subsequent iterations.
    b. Population based metaheuristic- start with more than one initial solution to start optimization.
2. Domain mimicked - most metaheuristic algorithms are named after natural phenomena
    a. Evolutionary algorithms(genetic algorithms, evolution strategies, differential evolution)
    b. Swarm intelligence algorithms.
    c. Physical phenomena - (simulated annealing)
    d. Unclear - Tabu search and scatter search

Classification

Other ways of classifying metaheuristic techniques:

1. Deterministic vs Stochastic methods - Deterministic follow a definite trajectory from the random initial solution. They are referred to as trajectory methods. Stochastic(discontinuous) allow for probabilistic jumps from current solution to the next.
2. Greedy vs non-greedy methods: Greedy search in the neighbourhood of the current solution and immediately move to a better solution when its found

Classification

3. Memory usage vs memoryless methods - memory based methods maintain a record of past solutions and their trajectories and use them to direct search.

4. One vs neighbourhood methods - some methods(tabu search and simulated annealing) allow a limited set of moves from the current solution.

5. Dynamic vs static objective function - methods that update the objective function depending on the current requirements.

Most metaheuristics are population based, stochastic, non-greedy and use a static objective function.

Generic Metaheuristic Framework

Most metaheuristic methods follow a similar sequence of operations and can be defined in a common generic framework.

1: Set iteration counter t = 0

2: Initialize N solutions St randomly

3: Evaluate each member of St

4: Mark the best solution of St as xt

5: repeat

6: Choose solutions (set Pt) from St using a selection plan (SP)

Generic Metaheuristic Framework

7: Generate new solutions (set Ct) from Pt using a generation plan (GP)

8: Choose solutions (set Rt) from St using a replacement plan (RP)

9: Update St by replacing Rt using solutions from a pool of any combination of at most three sets Pt, Ct, and Rt using an update plan (UP)

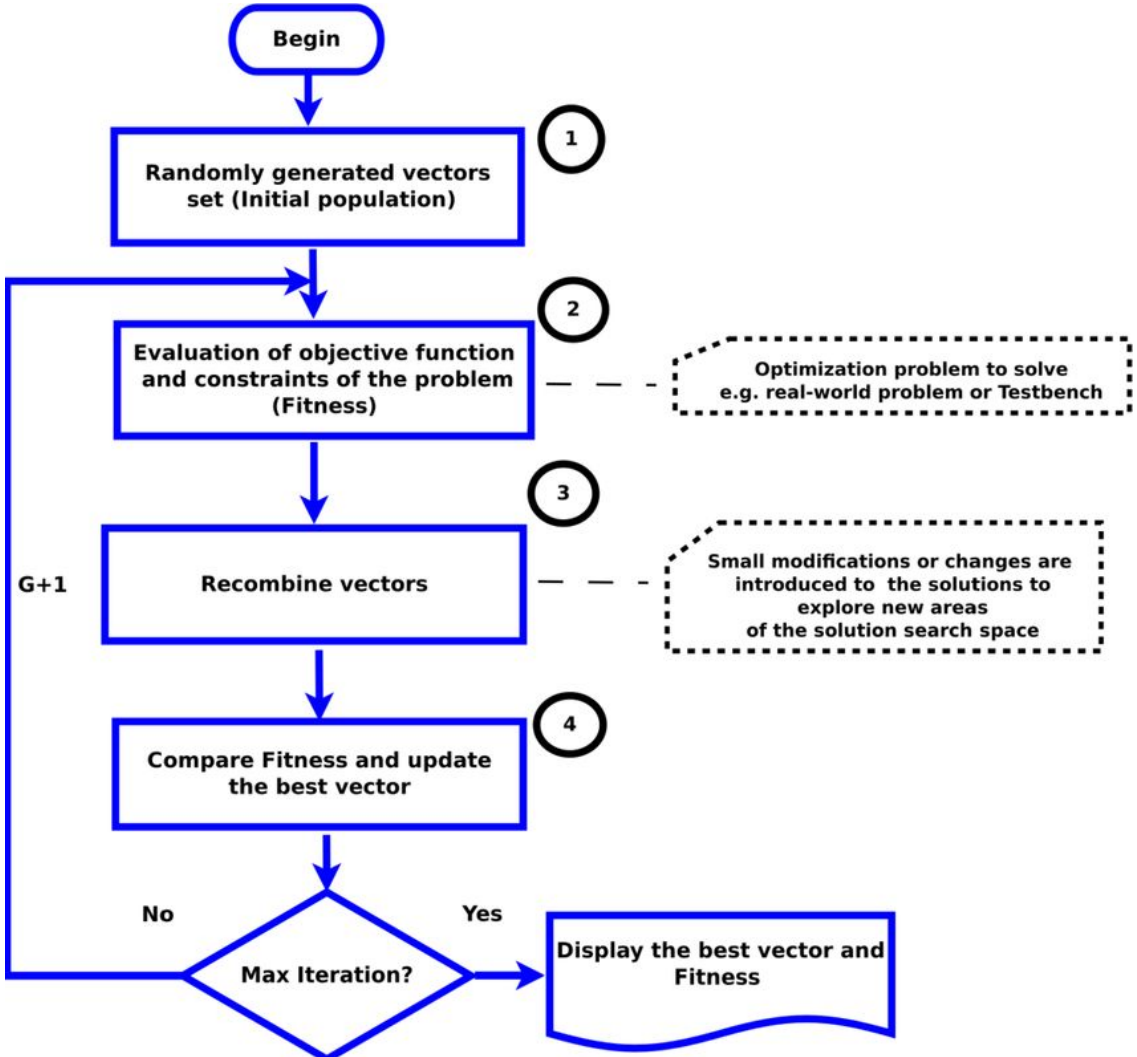
10. Evaluate each member of St

11. Identify the best solution of St and update xt

12. t <- t +1

13: until (a termination plan (TP) is satis ed)

14: Declare the near-optimal solution as xt


Begin
1
Randomly generated vectors set (Initial population)
2
Evaluation of objective function and constraints of the problem (Fitness)
Optimization problem to solve e.g. real-world problem or Testbench
3
Recombine vectors
Small modifications or changes are introduced to the solutions to explore new areas of the solution search space
G+1
4
Compare Fitness and update the best vector
No
Yes
Max Iteration?
Display the best vector and Fitness

Evolutionary Algorithms

Algorithms that reproduce essential elements of biological evolution to solve difficult problems to get approximate solutions for which exact solutions are not known.

Candidate solutions play the role of individuals in a population and the fitness function represents the quality of the solutions.

They perform well because they do not make assumptions on the underlying fitness landscape

Evolutionary Algorithms

Terms

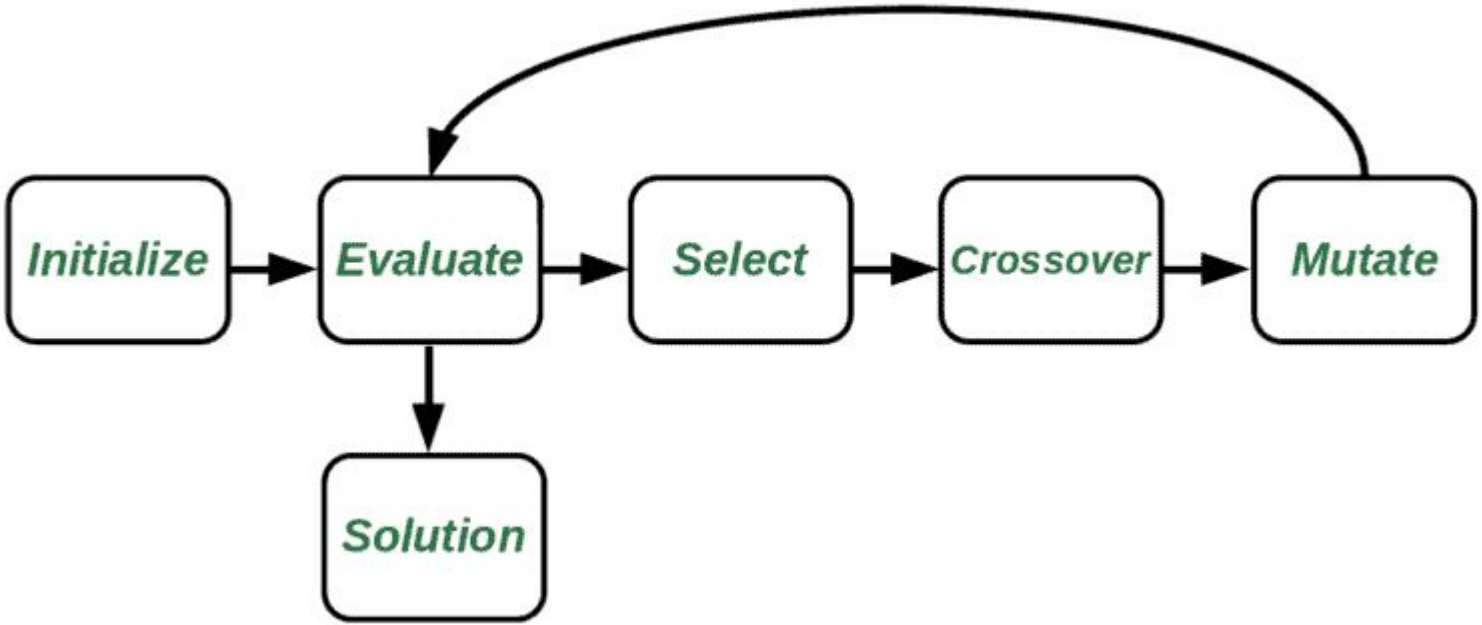
1. Population - Set of candidates for the solution.
2. Generation - step in the evolutionary process.
3. Selection - select best solutions to move to the next generation.
4. Crossover - exploitation of the different combination of individuals.
5. Mutation - Exploration of different environments.
6. Fitness Function - measures how close a solution is to the goal.

Evolutionary Algorithms

Generic Definition.

1. Randomly generate initial population of individuals on the first generation.
2. Evaluate fitness of each individual.
3. Check if the goal is reached and algorithm can be terminated.
4. Select individuals of higher fitness especially of higher fitness.
5. Produce offspring with optional **crossover**.
6. Apply **mutation**.
7. **Select** individuals of lower fitness score and replace with individuals of higher fitness score.

Evolutionary Algorithms



Genetic Algorithms

Genetic algorithms are adaptive metaheuristic search algorithms that belong to the larger part of evolutionary algorithms.

They are based on the general ideas of natural selection and genetics.

These are intelligent exploitation of random searches provided with historical data to direct the search into the region of better performance in solution space.

They are commonly used to generate high quality solutions for optimization problems and search problems.

Genetic Algorithms

They simulate the process of natural selection meaning that the species that adapt to changes in their environment can survive and reproduce and go to the next generation.

Each generation consists of a population of individuals and each individual represents a point in search space and possible solution.

Each individual is represented as a string of character/integer/floats/bits. This is analogous to the chromosome.

Genetic Algorithms

Foundations

1. Individuals in the population compete for resources and mate.
2. Those individuals who are successful (fittest) then mate to create more offspring than others.
3. Genes from the ‘fittest’ parent propagate throughout the generation, that is sometimes parents create better offspring.
4. Thus each successive generation is more suited for their environment.

Genetic Algorithms

Search Space

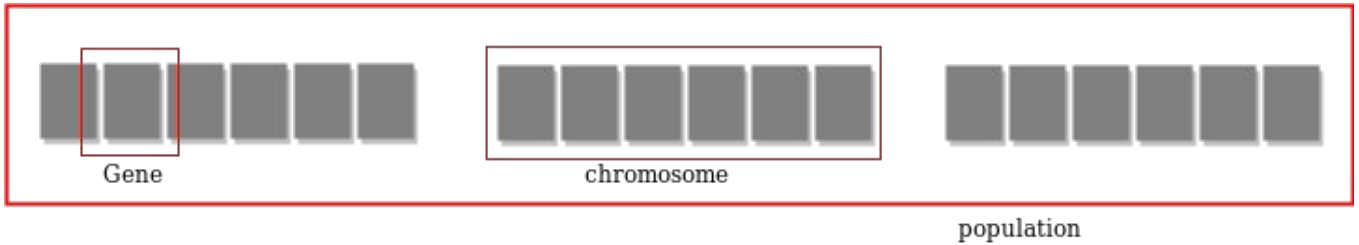
The population of individuals are maintained within search space.

Each individual represents a solution in search space for given problem.

Each individual is coded as a finite length vector (chromosomes) of components.

These variable components are analogous to genes.

Thus a chromosome (individual) is composed of several genes (variable components)



Genetic Algorithms

Fitness Score

A fitness score is given to each individual which shows the ability of an individual to compete. Individuals with optimal fitness score (or near optimal) are sought.

The algorithms maintain the population of n individuals(chromosomes/solutions) along with their fitness.

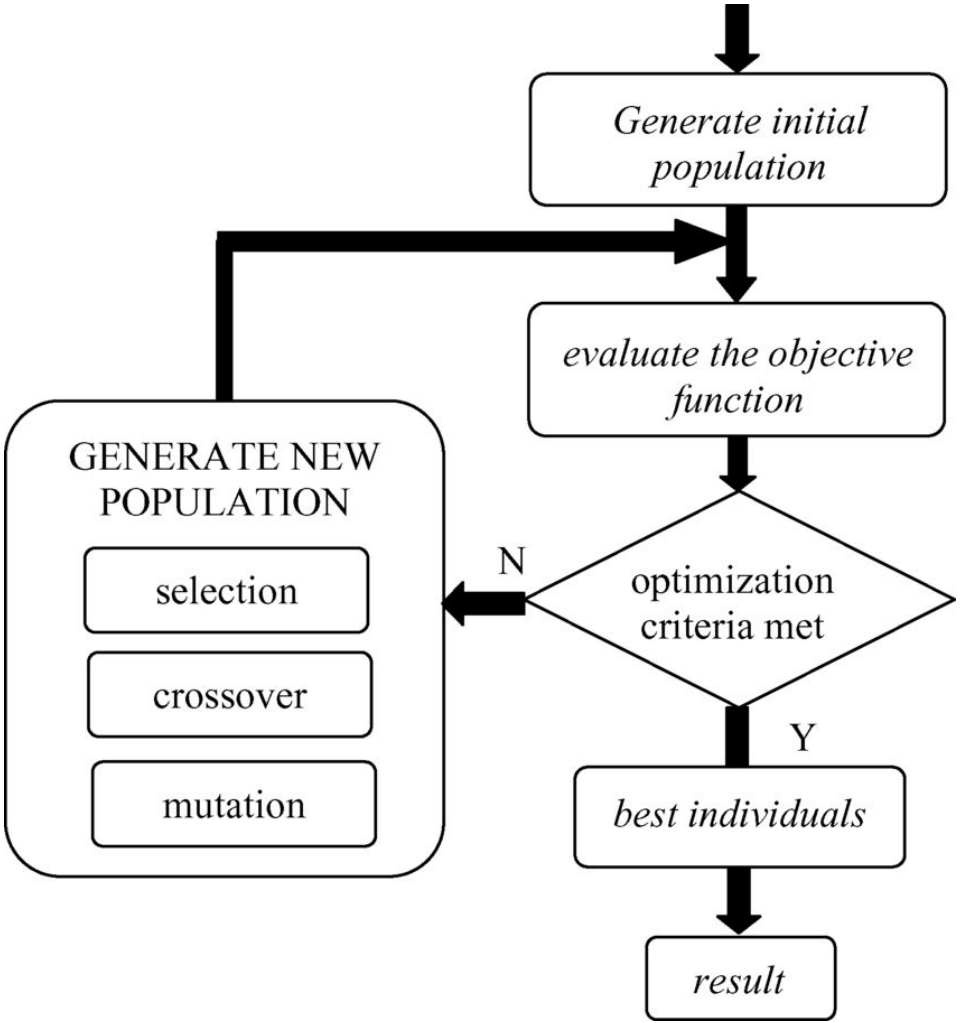
Individuals with a higher fitness score are given more chances to reproduce than others.

Each new generation has on average more “better genes” than the individual(solution) of previous generations. Thus each new generations have better “partial solution” than previous generation.

Genetic Algorithms

Operators of Genetic Algorithms

1. Selection Operator: The idea is to give preferences to the individuals with good fitness scores and allow them to pass their genes to successive generations.
2. Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are exchanged thus creating new offspring.
3. Mutation Operator: Insert random genes in offspring to maintain diversity.


Generate initial population
evaluate the objective function
optimization criteria met
N
Y
GENERATE NEW POPULATION
selection
crossover
mutation
best individuals
result

Genetic Algorithms

Example:

Using genetic algorithms for test case generation.

Genetic Algorithms

Why use Genetic Algorithms

- Robust
- Provide optimisation over large space state
- They don’t break on slight change in input or presence of noise

Applications

- Recurrent Neural Network
- Optimizing decision trees
- Mutation testing
- Code breaking
- Filtering and signal processing
- Learning fuzzy rule base etc

Genetic Algorithms

Advantages

- Efficiency
- Adaptability
- Optimization

Disadvantages

- Complexity
- Computational Cost

Particle Swarm Optimization

- Inspired by the flocking of birds and the schooling of fish.
- Its underlying mechanism allows particles to dynamically adjust their velocities based on personal and collective achievements, which results in a blend of individual effort and communal insight.
- PSO is a fantastic tool that can help you navigate complex functions and find global maxima with precision.

Particle Swarm Optimization

- PSO consists of a swarm of particles, each representing a prospective solution to an optimization problem. These particles explore the solution space by adjusting by adjusting their positions based on their own experiences and the successes of their neighbours.
- The goal is to find the global optimum of a given fitness function. This is achieved through movement of particles towards areas of the solution space that offer better outcomes as determined.

Particle Swarm Optimization

Key Components

- Each particle has a position, velocity and fitness value.
- The position represents a potential solution.
- The velocity indicates the direction and speed of movement.
- The fitness value represents how good the solution is.

Particle Swarm Optimization

How PSO works

Initialization

Particles are scattered randomly within the problem space, each representing a potential solution. This ensures a broad exploration from the outset.

Each particle is assigned a random velocity directing its movement in the search space. This velocity is crucial for the dynamic adjustment of particles' positions over iterations.

Particle Swarm Optimization

Iterative Optimization

Velocity Update: At each iteration, a particle's velocity is adjusted using the equation:

v[t+1] = w * v[t] + c1 * r1 * (pBest[t] — x[t]) + c2 * r2 * (gBest[t] — x[t])

This formula incorporates:

1. Inertia weight(w): Governs the particle's momentum balancing global and local exploration.
2. Cognitive Coefficient (c1): Reflects the particle's tendency to return to its personal best position, encouraging individual learning.
3. Social Coefficient (c2): Indicates the influence of the swarms' best known position encouraging individual learning
4. Position Update: Following the velocity update, the particle's position is updated to x[t+1] = x[t] + v[t+1], moving it closer to the optimal solution based on the newly calculated velocity.

Particle Swarm Optimization

Convergence Towards Optimal Solution

- The swarms collaborative movement, guided by both personal and global best positions, enables particles to explore and exploit problem space effectively.
- The process is repeated united stopping criteria is met, such as reaching a maximum number of iterations or finding a satisfactory solution.

Particle Swarm Optimization

Topology Types

1. Star (Global Best): Every particle is connected to every other particle, promoting rapid information dissemination but risking premature convergence.
2. Ring (Local Best): Particles are only connected to their immediate neighbours, slowing down convergence but enhancing exploration and prevents premature convergence.
3. Number of particles and iterations: The size of the swarm and the number of iterations are crucial parameters that need to be optimized for each problem

Particle Swarm Optimization

Adaptive Strategies

- Self tuning mechanisms: Given the sensitivity of PSO's performance to its parameters, self-tuning strategies that adjust parameters based on feedback from the optimization process.
- Implementation: This involves dynamically changing the inertia weight or the cognitive and social coefficients in the response to the current state of the search, e.g rate of convergence or the diversity of the solutions within the swarm.

Particle Swarm Optimization

Advantages

- PSO is flexible and easy to implement algorithm that doesn’t require hyperparameter tuning.

Applications

- Neural Network Training
- Optimization of electric power distribution networks
- Structural optimization
- System identification in biomechanics

Particle Swarm Optimization

Challenges

- Slow convergence during refined search stage
- Weaker local search capability

Simulated Annealing

- Inspired by the physical process of annealing in metallurgy, Simulated Annealing is a probabilistic technique used for solving both combinatorial and continuous optimization problems.
- Designed to search for an optimal or near optimal in a large solution space.
- The name and concept are derived from the process of annealing in metallurgy, where a material is heated and then slowly cooled to remove defects and achieve a stable crystalline structure.
- The ‘heat’ corresponds to the degree of randomness in the search process, which decreases over time(cooling) to refine the solution.
- Used widely in combinatorial optimization where problems often have numerous local optima

Simulated Annealing

How it works

It starts with an initial solution and a high ‘temperature’ which gradually decreases over time.

Steps:

- Initialization: Begin with an initial solution So and an initial temperature To. The temperature controls how likely the algorithm is to accept worse solutions as it explores search space.
- Neighbourhood search: At each step, new solution S is generated by making a small change (or perturbation) to the current solution S.
- Objective Function Evaluation: The new solution S is evaluated using the objective function. I f S provides a better solution than S, it is accepted as a new solution.
- Acceptance Probability: If S’ is worse than S, it may still be accepted with a probability based on temperature and difference in objective function values.
- Cooling Schedule: After each iteration, the temperature is decreased according to a specific cooling schedule.
- Termination: The algorithm continues until the system a low system

Simulated Annealing

Cooling Procedure

- Cooling procedure plays a crucial role in the performance of Simulated Annealing.
- If the temperature decreases too quickly, the algorithm may converge prematurely to a suboptimal solution (local optimum).
- On the other end if the cooling is too slow the algorithm may take a long time to find the optimal solution

Simulated Annealing

Advantages

- Ability to escape local minima:The probabilistic acceptance of worse solutions allows algorithms to explore a broader solution space.
- Simple Implementation: The algorithm is relatively easy to implement and can be adapted to a wide range of optimization problems.
- Global Optimization: It can approach a global optimum, especially when paired with a well designed cooling schedule.

Simulated Annealing

Limitations

- Parameter Sensitivity: Performance is dependent on the choice of parameters, particularly the initial temperature and cooling schedule.
- Computational Time: It can be computationally expensive for large problems.
- Slow Convergence: slower than other methods

Simulated Annealing

Applications

- Traveling Salesman Problem (TSP): In combinatorial optimization, it is often used to find near-optimal solutions for the TSP, where a salesman must visit a set of cities and return to the origin, minimizing the total travel distance.
- VLSI Design: SA is used in the physical design of integrated circuits, optimizing the layout of components on a chip to minimize area and delay.
- Machine Learning: In machine learning, SA can be used for hyperparameter tuning, where the search space for hyperparameters is large and non-convex.
- Scheduling Problems: SA has been applied to job scheduling, minimizing delays and optimizing resource allocation.
- Protein Folding: In computational biology, SA has been used to predict protein folding by optimizing the conformation of molecules to achieve the lowest energy state.

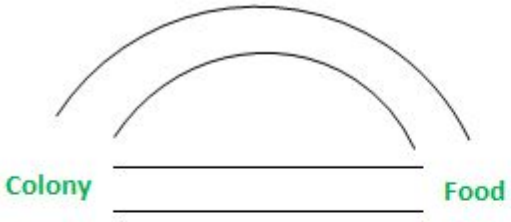
Ant Colony Optimization

Inspired from the foraging behavior of ant colonies. They communicate using pheromones

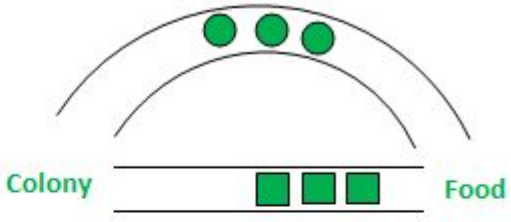
ACO observes the movement of ants as they search for food in the shortest possible path.

Initially ants start to move randomly in search of food around their nests, this opens multiple routes from the nests to the food source/

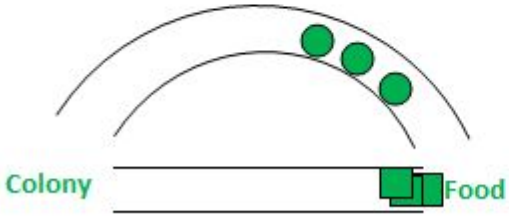
Based on the quality and quantity of the food, ants carry a portion back with necessary pheromone concentration as well as the rate of evaporation of the pheromone.



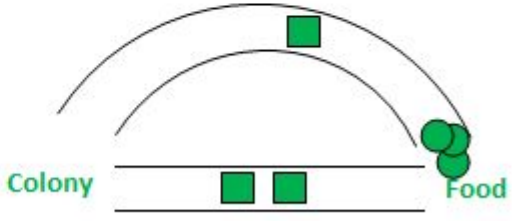
1



2



3



4

Ant Colony Optimization

Stage 1: All ants are in their nest. There is no pheromone content in the environment. (For algorithmic design, residual pheromone amount can be considered without interfering with the probability)

Stage 2: Ants begin their search with equal (0.5 each) probability along each path. Clearly, the curved path is the longer and hence the time taken by ants to reach food source is greater than the other.

Stage 3: The ants through the shorter path reaches food source earlier. Now, evidently they face with a similar selection dilemma, but this time due to pheromone trail along the shorter path already available, probability of selection is higher.

Stage 4: More ants return via the shorter path and subsequently the pheromone concentrations also increase. Moreover, due to evaporation, the pheromone concentration in the longer path reduces, decreasing the probability of selection of this path in further stages. Therefore, the whole colony gradually uses the shorter path in higher probabilities. So, path optimization is attained.

Ant Colony Optimization

Algorithm Design

We consider a single food source, single ant colony and 2 paths for traversal.

The whole scenario can be realized through weighted graphs where the ant colony and the food source act as vertices (or nodes); the paths serve as the edges and the pheromone values are the weights associated with the edges.

Ant Colony Optimization

Pseudocode

```
Procedure AntColonyOptimization:
    Initialize necessary parameters and pheromone trials;
    while not termination do:
        Generate ant population;
        Calculate fitness values associated with each ant;
        Find best solution through selection methods;
        Update pheromone trial;
    end while
end procedure
```

Tabu Search

- Tabu search is a local search algorithm that aims to escape local optima by maintaining a tabu list (short term memory of recently visited solutions)
- The algorithm explores the solution space by moving from the current solution to its best neighbouring solution, even if it leads to a worse objective function value.
- To prevent cycling back to previously visited solutions, the tabu list forbids for a specific number of iterations (tabu tenure).

Tabu Search

How it works

- It starts with an initial solution and iteratively it by applying local search moves.
- At each iteration, the algorithm generates a set of candidate solutions in the neighbourhood of the current solution, even if it’s worse than the previous one.
- If the selected move is in the tabu list, it is only accepted, it is only accepted if it satisfies an aspiration criterion, allowing the algorithm to override the tabu status of a move if it leads to as solution better than the best solution found so far.

Tabu Search

- It also incorporates intensification and diversification strategies to balance the exploration and exploitation of the solution space.
- Intensification focuses on exploring promising regions of the solution space more thoroughly while diversification encourages the algorithm to explore new regions and escape from local optima.

Tabu Search

Procedure

1. Initialize:
    - Generate an initial solution s
    - Set the best solution s_best = s
    - Initialize the tabu list to be empty
    - Set the tabu tenure t
    - Set the maximum number of iterations max_iter
    - Set the current iteration counter iter = 0

Tabu Search

Procedure

2. While iter < max_iter:
    - Generate a set of candidate solutions N(s) in the neighborhood of s
    - Select the best candidate solution s’ from N(s) that is not in the tabu list or satisfies the aspiration criterion
    - Update the best solution: if f(s’) < f(s_best), set s_best = s′
    - Update the current solution: s = s′
    - Update the tabu list:
        - Add the move that generated s’ to the tabu list
        - If the tabu list size exceeds the tabu tenure t, remove the oldest move from the list
    - Increment the iteration counter: iter = iter + 1
3. Return the best solution found s_best

Tabu Search

Data Structures

- Solution: Represents a candidate solution in the solution space
- Tabu List: A list that stores the recently visited solutions or moves
- Aspiration Criterion: A condition that allows the algorithm to override the tabu status of a move

Parameters

- Tabu Tenure: The number of iterations for which a move or solution remains in the tabu list
- Maximum Iterations: The maximum number of iterations the algorithm will run before terminating
- Neighborhood Size: The number of candidate solutions generated in each iteration

Tabu Search

Advantages

- Tabu Search can escape local optima by allowing moves to worse solutions and preventing cycling back to previously visited solutions
- The tabu list provides a short-term memory mechanism that helps the algorithm explore new regions of the solution space
- Tabu Search is flexible and can be easily adapted to various optimization problems

Disadvantages

- The performance of Tabu Search depends on the appropriate setting of the tabu tenure and other parameters
- The algorithm may require a large number of iterations to converge to a high-quality solution
- Tabu Search may struggle with highly constrained problems or problems with a large number of local optima

ThankYou!