

21CS2204RA – Mathematical Programming
A Project Report
on
TRANSPORTATION PROBLEM BY NW METHOD
and
ANT COLONY OPTIMIZATION

Under the Guidance of
Dr. Sanasam Inunganbi
Associate Professor, Department of CSE-H

by		
I.D NUMBER	NAME	SECTION
2100030034	B. Rohanth	12

KONERU LAKSHMAIAH EDUCATION FOUNDATION
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(DST-FIST Sponsored Department)
Green Fields, Vaddeswaram, Guntur District-522 502

April - 2023

Declaration

We here by declare that this Project report entitled “**TRANSPORTATION PROBLEM BY NW METHOD**” and “**ANT COLONY OPTIMIZATION**” has been prepared by us in the course **21CS2204RA Mathematical Programming** in **COMPUTER SCIENCE AND ENGINEERING** during the Even Semester of the academic year 2022-2023. We also declare that this project-based lab report is of our own effort.

Date:

Place:

Signature of the Student

Student Name

Id Number

B. Rohanth

2100030034

CERTIFICATE

This is to certify that the project based Lab report entitled “**ANT COLONY OPTIMIZATION**” and “**TRANSPORTATION PROBLEM BY NW METHOD**” is a bonafide work done Mr. B. Rohanth bearing Regd.No.**2100030034** to the course **21CS2204RA Mathematical Programming** in COMPUTER SCIENCE AND ENGINEERING during the Even Semester of Academic year 2022-2023.

FACULTY IN CHARGE

Dr. Sanasam Inunganbi

ACKNOWLEDGEMENTS

Our sincere thanks to Dr. Sanasam Inunganbi in the lab sessions for her outstanding support throughout the project for the successful completion of the work.

We express our gratitude to Dr. Sanasam Inunganbi, Course Co-Ordinator for the course 21CS2204AA Mathematical Programming in the Department of Computer Science and Engineering for providing us with adequate planning and support and means by which we can complete this project.

We express our gratitude to **Prof. A. Senthil**, Head of the Department for Computer Science and Engineering for providing us with adequate facilities, ways and means by which we can complete this project.

We would like to place on record the deep sense of gratitude to the Vice Chancellor, K L University for providing the necessary facilities to carry out the project.

Last but not the least, we thank all Teaching and Non-Teaching Staff of our department and especially our classmates and our friends for their support.

Name: B. Rohanth

Reg. No: 2100030034

TRANSPORTATION PROBLEM BY NW METHOD INDEX

S.NO	TITLE	PAGE NO
1	Abstract	7
2	Introduction	8
3	Notes	9 - 10
4	Objective	11
5	Advantages	12
6	Disadvantages	13
7	Procedure/Algorithm	14 - 18
8	Pseudo code	19
9	Python Code	20 - 21
10	Conclusion	22

ANT COLONY OPTIMIZATION INDEX

S.NO	TITLE	PAGE NO
1	Abstract	23
2	Introduction	24
3	Notes	25 - 28
4	Objective	29
5	Advantages	30
6	Disadvantages	31
7	Procedure/Algorithm	32 - 33
8	Pseudo code	34
9	Python Code	35 - 38
10	Conclusion	39

1. TRANSPORTATION PROBLEM BY NW METHOD

ABSTRACT

The Northwest Corner Rule (NWCR) is a basic method used in solving transportation problems. In this method, allocation of supply is started at the top left corner (i.e., the northwest corner) of the transportation table and then the algorithm progresses by filling up the next available supply cell in the same row until the supply is exhausted. After that, the algorithm moves to the next row and continues the allocation process until all supply and demand are fulfilled. The solution obtained by this method is usually not optimal, but it provides a good starting point for further optimization methods.

The NWCR method is simple to understand and implement, making it a popular choice for beginners in transportation problem solving. However, it may not always provide an optimal solution and can be quite inefficient when the problem size is large. Therefore, it is often used as a starting point for more sophisticated methods that provide a better solution.

In summary, the NWCR is an easy and intuitive method to use when solving transportation problems. It involves starting at the top-left corner of the transportation table and then progressing row by row, filling up the next available supply cell until the supply is exhausted. While the solution may not be optimal, it provides a good starting point for further optimization.

INTRODUCTION

The transportation problem is a classical optimization problem that deals with finding the least cost solution to transport a commodity from various sources to various destinations. The objective is to minimize the total transportation cost while meeting the supply and demand requirements.

The Northwest Corner (NW) Method is a popular method for solving the transportation problem. It is a heuristic approach that starts by allocating shipments from the northwest corner of the transportation matrix and iteratively moving to the next cell along the row or column with the smallest cost until all supply and demand requirements are met.

The NW Method works by constructing an initial feasible solution by starting at the northwest corner of the transportation matrix, assigning as much as possible to the available capacity in that cell, and then adjusting the supply and demand accordingly. The algorithm then moves to the next cell in the same row or column and repeats the process until all supply and demand requirements are met.

The NW Method is simple to understand and implement, making it a popular method for small to medium-sized transportation problems. However, it may not always result in the optimal solution and can be slow for larger problems. Other methods such as the Least Cost Method, Vogel's Approximation Method, and the Modified Distribution Method can also be used to solve transportation problems.

NOTES

Here are some key points to keep in mind when solving the transportation problem using the NW Method:

1. The transportation problem involves minimizing the total transportation cost while meeting the supply and demand requirements.
2. The NW Method is a heuristic approach that starts by allocating shipments from the northwest corner of the transportation matrix and iteratively moving to the next cell along the row or column with the smallest cost until all supply and demand requirements are met.
3. The NW Method constructs an initial feasible solution by starting at the northwest corner of the transportation matrix, assigning as much as possible to the available capacity in that cell, and then adjusting the supply and demand accordingly.
4. The algorithm then moves to the next cell in the same row or column and repeats the process until all supply and demand requirements are met.
5. The NW Method may not always result in the optimal solution and can be slow for larger problems.
6. To use the NW Method, first create a transportation matrix with the supply and demand values and the transportation costs.

7. Starting at the northwest corner of the matrix, allocate as much as possible to the available capacity in that cell.
8. Adjust the supply and demand accordingly, and then move to the next cell in the same row or column with the smallest cost.
9. Continue this process until all supply and demand requirements are met.
10. Once an initial feasible solution is obtained using the NW Method, it can be improved using other optimization techniques like the Modified Distribution Method or the Vogel's Approximation Method.

OBJECTIVE

The objective of the transportation problem by the NW Method is to find the least cost solution for transporting a commodity from various sources to various destinations while meeting the supply and demand requirements. The goal is to minimize the total transportation cost, which is the sum of the cost of transporting the commodity from each source to each destination, subject to the supply and demand constraints. The NW Method is one approach for finding an initial feasible solution to the transportation problem, but it may not always result in the optimal solution. The final objective is to find the optimal solution, which can be achieved by using other optimization techniques to improve the initial feasible solution obtained using the NW Method.

- To allocate the available resources (supply) to various destinations (demand) in the most efficient way possible.
- To minimize the total transportation cost, including the cost of shipping the commodity from each source to each destination and any fixed costs associated with transportation.
- To ensure that the supply and demand requirements are met and that no excess or shortage occurs during transportation.

ADVANTAGES

- The NW Method is a simple and easy-to-understand heuristic approach that can be used to obtain an initial feasible solution to the transportation problem.
- The method is relatively fast and efficient for small to medium-sized transportation problems.
- The NW Method provides a good starting point for more complex optimization techniques that can be used to improve the initial feasible solution.
- The method is applicable to a wide range of transportation problems, including those with multiple sources and destinations.
- The NW Method can be used to identify bottlenecks or inefficiencies in the transportation network, which can then be addressed using other optimization techniques.
- The method is useful for decision-making in industries such as manufacturing, logistics, and supply chain management, where transportation is a key factor in the business operations.
- The NW Method is flexible and can be adapted to different scenarios, allowing for customization to specific business needs.

DISADVANTAGES

- The NW Method may not always result in the optimal solution, particularly for larger and more complex transportation problems.
- The method relies heavily on the initial allocation of shipments from the northwest corner of the transportation matrix, which can sometimes result in a suboptimal solution.
- The NW Method does not consider other factors that may affect the transportation process, such as time constraints, capacity limitations, and route restrictions.
- The method assumes that the transportation costs are fixed and does not take into account any variable costs that may arise during transportation.
- The NW Method may not be suitable for transportation problems with multiple modes of transportation, such as air, sea, and land transportation.
- The method assumes that the supply and demand requirements are known and fixed, which may not always be the case in real-world transportation scenarios.

PROCEDURE / ALGORITHM

1. Select the upper left (north-west) cell of the transportation matrix and allocate minimum of supply and demand, i.e., $\min(A_1, B_1)$ value in that cell.
2. Obtaining Optimal Basic Solution.
 - If $A_1 < B_1$, then allocation made is equal to the supply available at the first source (A_1 in first row), then move vertically down to the cell (2,1).
 - If $A_1 > B_1$, then allocation made is equal to demand of the first destination (B_1 in first column), then move horizontally to the cell (1,2).
 - If $A_1 = B_1$, then allocate the value of A_1 or B_1 and then move to cell (2,2).
3. Continue the process until an allocation is made in the south-east corner cell of the transportation table.

Example :

In the table, three sources A,B and C with the Production Capacity of 50units, 40units, 60 units of product respectively is given. Every day the demand of three retailers P,Q,R is to be furnished with at least 20units, 95units and 35units of product respectively. The transportation costs are also given in the matrix.

Source/Destination	P	Q	R	Supply
A	5	8	4	50
B	6	6	3	40
C	3	9	6	60
Demand	20	95	35	150

Sol : TRANSPORTATION PROBLEM USING NW METHOD

Step – 1 :

Check whether Total Demand is equal to Total Supply. In case the demand is more then supply, then dummy origin is added to the table. The cost associated with the dummy origin will be zero.

Step – 2 :

North-west cell is (1,1). Allocate $\min\{20,50\}$ to North-west cell (1,1). Update the demand and supply.

Source/Destination	P	Q	R	Supply
A	20 5	8	4	50 30
B	6	6	3	40
C	3	9	6	60
Demand	20	95	35	150
	0			

Step – 3 :

Next north-west cell is (1,2). Allocate $\min\{95,30\}$ to the cell (1,2). Update the demand and supply.

Source/Destination	P	Q	R	Supply
A	20 5	30 8	4	50 30 0
B	6	6	3	40
C	3	9	6	60
Demand	20	95	35	150
	0	65		

Step – 4 :

Next north-west cell is (2,2). Allocate $\min\{65,40\}$ to the cell (2,2). Update the demand and supply.

Source/Destination	P	Q	R	Supply
A	20 5	30 8	4	50 30 0
B	6	40 6	3	40 0
C	3	9	6	60
Demand	20	95	35	150
	0	65 25		

Step – 5 :

Next north-west cell is (3,2). Allocate $\min\{25,60\}$ to the cell (3,2). Update the demand and supply.

Source/Destination	P	Q	R	Supply	
A	20 5	30 8	4	50	30 0
B	6	40 6	3	40	0
C	3	25 9	6	60	35
Demand	20	95	35	150	
	0	65 25 0			

Step – 6 :

Next north-west cell is (3,3). Allocate $\min\{35,35\}$ to the cell (3,3). Update the demand and supply.

Source/Destination	P	Q	R	Supply	
A	20 5	30 8	4	50	30 0
B	6	40 6	3	40	0
C	3	25 9	35 6	60	35 0
Demand	20	95	35	150	
	0	65 25 0	0		

Total Cost can be computed by multiplying the units assigned to each cell with the concerned transportation cost.

$$\begin{aligned} \text{Total Cost} &= (20 * 5) + (30 * 8) + (40 * 6) + (25 * 9) + (35 * 6) \\ &= \text{Rs.1015.} \end{aligned}$$

Total Cost = Rs. 1015

PSEUDO CODE

1. Initialize the transportation matrix with supply and demand values and transportation costs.
2. Set $i = 1$ and $j = 1$ to start at the northwest corner of the matrix.
3. While supply and demand requirements are not met:
 - a. Allocate as much as possible to the available capacity in the current cell.
 - b. Adjust the supply and demand accordingly.
 - c. If supply is exhausted, move to the next row ($i++$) and start at the first column ($j=1$).
 - d. If demand is satisfied, move to the next column ($j++$) and start at the current row.
4. Compute the total transportation cost for the initial feasible solution.
5. Improve the solution using other optimization techniques if necessary.

IMPLEMENTATION OF PYTHON PROGRAM

```
import numpy as np

supply = np.array([50, 40, 60])
demand = np.array([20, 90, 55])

cost = np.array([[5, 8, 4],
                 [6, 6, 3],
                 [3, 9, 6]])
allocation = np.zeros((3, 3))

# North West Corner Rule
for i in range(len(supply)):
    for j in range(len(demand)):
        allocation[i][j] = min(supply[i], demand[j])
        supply[i] -= allocation[i][j]
        demand[j] -= allocation[i][j]

print("Initial Feasible Solution:")
print(allocation)
total_cost = 0
for i in range(len(allocation)):
    for j in range(len(allocation[i])):
        total_cost += allocation[i][j] * cost[i][j]

print("Total Cost: ", total_cost)
```

OUTPUT –

Initial Feasible Solution:

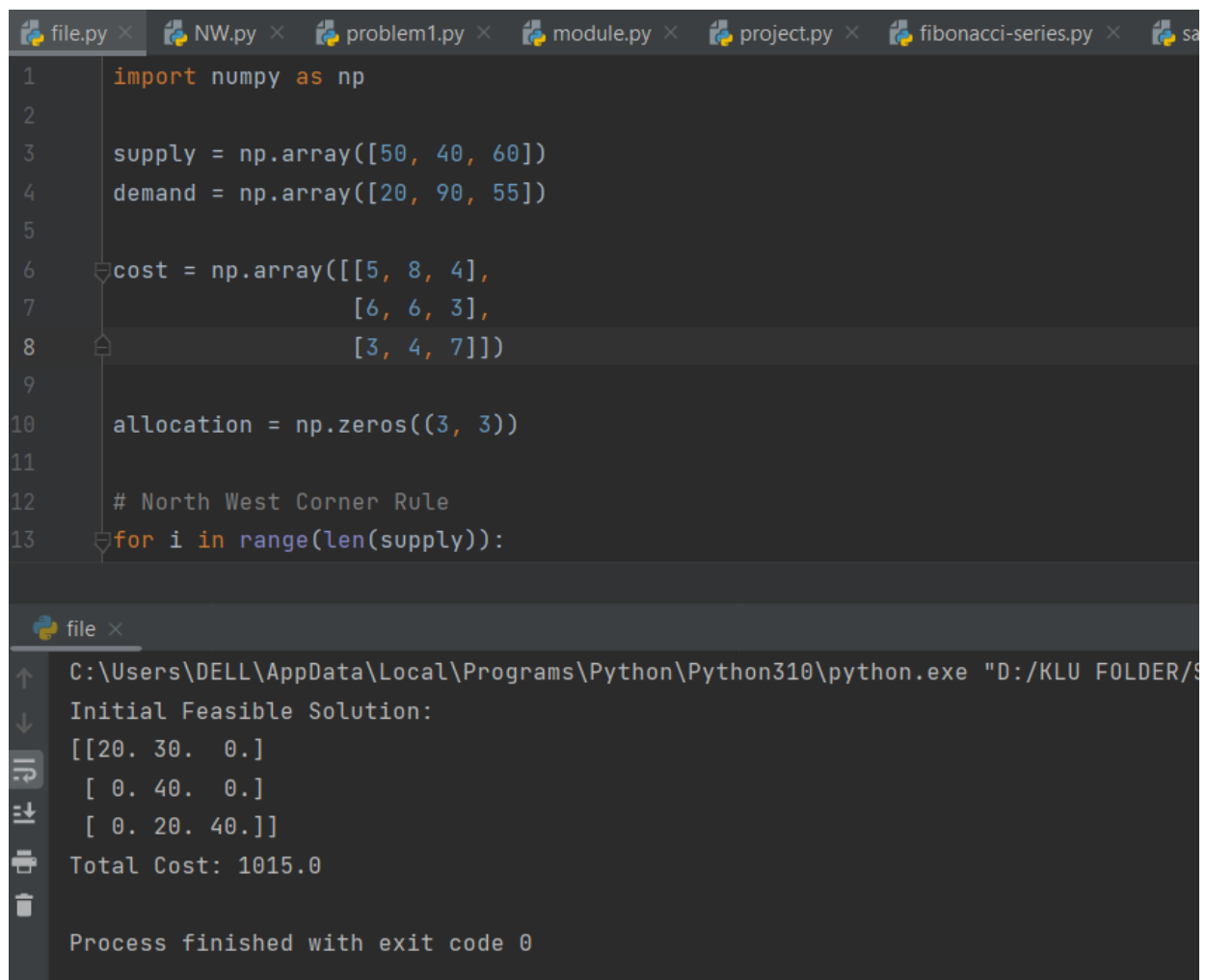
[[20. 30. 0.]

[0. 40. 0.]

[0. 20. 40.]]

Total Cost: 1015.0

EXECUTION –



```
1 import numpy as np
2
3 supply = np.array([50, 40, 60])
4 demand = np.array([20, 90, 55])
5
6 cost = np.array([[5, 8, 4],
7                  [6, 6, 3],
8                  [3, 4, 7]])
9
10 allocation = np.zeros((3, 3))
11
12 # North West Corner Rule
13 for i in range(len(supply)):
```

file x

C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe "D:/KLU FOLDER/S

Initial Feasible Solution:

[[20. 30. 0.]

[0. 40. 0.]

[0. 20. 40.]]

Total Cost: 1015.0

Process finished with exit code 0

CONCLUSION

The Least Cost Method is considered to produce more optimal results than the North-west Corner because it considers the cost while making the allocation, whereas The Least Cost Method is another method used to obtain the initial feasible solution for the transportation problem. Here, the allocation begins with the cell which has the minimum cost. The lower cost cells are chosen over the higher-cost cell with the objective to have the least cost of transportation

2. ANT COLONY OPTIMIZATION

ABSTRACT

Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the behavior of ants searching for food. The algorithm is based on the concept of pheromone trail communication among ants and their ability to follow the most promising trail to the food source. ACO has been applied to a wide range of optimization problems, including the Traveling Salesman Problem, Vehicle Routing Problem, and Job Shop Scheduling Problem. The algorithm has shown to be effective in finding near-optimal solutions for these problems.

ACO is a population-based algorithm that uses a probabilistic approach to construct solutions, and it has the advantage of being able to handle large-scale combinatorial problems with a large number of variables. ACO has also been extended and modified to address various limitations, such as premature convergence and parameter tuning.

In ACO, a colony of artificial ants is used to search for the optimal solution to a given problem. The ants communicate with one another by laying down pheromone trails, which represent the quality of the solutions that they have found. The other ants use these pheromone trails to guide their search, with a higher concentration of pheromone indicating a better solution.

ACO has been applied to a wide range of problems in different fields, including engineering, transportation, and logistics. One of the most well-known applications of ACO is in solving the Traveling Salesman Problem, where the goal is to find the shortest route that visits all given cities and returns to the starting point.

INTRODUCTION

ACO uses a similar approach to solve optimization problems. The algorithm starts with a population of artificial ants that search for the optimal solution to a given problem. The ants construct solutions by selecting a path through a solution space, guided by the concentration of pheromone on each path. The pheromone concentration on each path is updated based on the quality of the solutions found by the ants. This pheromone update rule encourages the ants to focus their search on the most promising solutions, increasing the probability of finding the global optimum.

ACO is a population-based algorithm that uses a probabilistic approach to construct solutions. The algorithm has the advantage of being able to handle large-scale combinatorial problems with a large number of variables. ACO has been successfully applied to a variety of optimization problems, including the Traveling Salesman Problem, Vehicle Routing Problem, and Job Shop Scheduling Problem.

The success of ACO has led to the development of many variations and extensions of the algorithm. These variations include modifications to the pheromone update rule, the addition of local search algorithms, and the use of multiple colonies of ants. The flexibility of ACO makes it a powerful tool for solving optimization problems in different fields, such as engineering, transportation, and logistics.

ACO has been applied to a wide range of problems in different fields, including engineering, transportation, and logistics. One of the most well-known applications of ACO is in solving the Traveling Salesman Problem, where the goal is to find the shortest route that visits all given cities and returns to the starting point.

NOTES

- ACO is a metaheuristic algorithm that is inspired by the foraging behavior of ants.
- ACO uses a population of artificial ants to search for the optimal solution to a given problem.
- Ants construct solutions by selecting a path through a solution space, guided by the concentration of pheromone on each path.
- The pheromone concentration on each path is updated based on the quality of the solutions found by the ants.
- ACO is a probabilistic approach that is able to handle large-scale combinatorial problems with a large number of variables.
- ACO has been successfully applied to a variety of optimization problems, including the Traveling Salesman Problem, Vehicle Routing Problem, and Job Shop Scheduling Problem.
- ACO has many variations and extensions, including modifications to the pheromone update rule, the addition of local search algorithms, and the use of multiple colonies of ants.
- ACO is a powerful tool for solving optimization problems in different fields, such as engineering, transportation, and logistics.

- ACO is a type of swarm intelligence, where a population of agents collectively solves a problem.
- ACO is a constructive approach, where the solution is built up step-by-step through the actions of the agents.
- The pheromone trails left by the ants provide a form of indirect communication between the agents, allowing them to share information about the quality of different solutions.
- ACO is a global optimization algorithm, meaning that it searches for the global optimum rather than a local optimum.
- ACO can be computationally expensive, especially for large-scale problems or problems with many variables.
- ACO can be parallelized to improve its efficiency, either by using multiple processors or by using a distributed computing system.
- ACO has been used in a variety of real-world applications, such as optimizing bus routes, scheduling aircraft maintenance, and designing telecommunications networks.
- ACO is a robust algorithm that is able to handle noisy or uncertain data, making it useful in situations where other optimization techniques may not work well.

- ACO is a nature-inspired algorithm that is based on the observation of real ant colonies and their behavior.
- ACO can be used to solve a wide range of optimization problems, including problems with discrete and continuous variables.
- ACO can be combined with other optimization techniques, such as genetic algorithms or simulated annealing, to improve its performance.
- ACO can be used to find near-optimal solutions quickly, making it useful in situations where solutions need to be found in a short amount of time.
- ACO can be sensitive to the choice of parameters, such as the amount of pheromone deposited and the evaporation rate of the pheromone.
- ACO can be adapted to incorporate domain-specific knowledge, such as heuristic information or constraints, to improve its performance on specific problems.
- ACO has been used in multi-objective optimization, where the goal is to optimize multiple objectives simultaneously, by using different types of pheromones to represent different objectives.

- ACO can be used in dynamic optimization problems, where the problem changes over time, by adapting the pheromone update rule to reflect changes in the problem.
- ACO is a popular optimization technique and has been used in a wide range of fields, including engineering, computer science, operations research, and biology.
- ACO can be used to solve problems that are difficult or impossible to solve using exact optimization techniques, such as integer programming or dynamic programming.
- ACO can be used to find high-quality solutions to problems that have multiple local optima, by using the pheromone trails to guide the search towards the global optimum.
- ACO has been used to solve complex problems, such as protein folding and circuit layout optimization, where the solution space is extremely large.
- ACO can be used to find robust solutions that are insensitive to small changes in the problem parameters or input data, making it useful in real-world applications.
- ACO can be used to solve problems in real-time, such as real-time traffic routing, by continuously updating the pheromone trails and re-evaluating the solutions.

OBJECTIVE

The objective of Ant Colony Optimization (ACO) is to find the optimal solution to a given optimization problem, inspired by the behavior of real ant colonies.

ACO achieves this objective by using a population of artificial ants that construct solutions by selecting a path through a solution space, guided by the concentration of pheromone on each path. The pheromone concentration on each path is updated based on the quality of the solutions found by the ants, with the goal of guiding the search towards the global optimum. ACO is a metaheuristic algorithm that is able to handle large-scale combinatorial problems with a large number of variables, and has been successfully applied to a variety of optimization problems, including the Traveling Salesman Problem, Vehicle Routing Problem, and Job Shop Scheduling Problem. The ultimate objective of ACO is to provide an effective and efficient optimization tool that can be used in a wide range of fields, such as engineering, transportation, and logistics, to improve decision-making and problem-solving.

ADVANTAGES

- **Robustness:** ACO is a robust optimization technique that is able to handle noisy or uncertain data. This makes it useful in situations where other optimization techniques may not work well.
- **Global Optimization:** ACO is a global optimization algorithm, meaning that it searches for the global optimum rather than a local optimum. This makes it useful in problems where finding the best solution is critical.
- **Adaptability:** ACO is an adaptable algorithm that can be modified to suit different types of problems. This makes it a versatile tool for optimization.
- **Nature-Inspired:** ACO is a nature-inspired algorithm that is based on the observation of real ant colonies and their behavior. This makes it an interesting and innovative technique for solving optimization problems.
- **Efficiency:** ACO is an efficient optimization technique that can find near-optimal solutions quickly. This makes it useful in situations where solutions need to be found in a short amount of time.
- **Scalability:** ACO can be scaled up to handle large-scale optimization problems. This makes it useful in problems with a large number of variables or complex solution spaces.

DISADVANTAGES

- **Parameter Tuning:** ACO has several parameters that need to be tuned in order to achieve good performance, such as the pheromone evaporation rate and the pheromone update rate. Finding the optimal values for these parameters can be time-consuming and requires some trial and error.
- **Premature Convergence:** ACO can converge prematurely to a local optimum if the pheromone trails are too strong, which can lead to suboptimal solutions. This can be mitigated by introducing diversity into the search process.
- **Sensitivity to Initial Conditions:** ACO is sensitive to the initial conditions of the pheromone trails, which can affect the quality of the solutions found. This can be mitigated by using a random initialization process or by using a heuristic to initialize the pheromone trails.
- **Computational Complexity:** ACO can be computationally expensive for large-scale optimization problems, as the search space can be very large. This can be mitigated by using parallelization or other optimization techniques in conjunction with ACO.
- **Limited to Discrete Problems:** ACO is mainly used for discrete optimization problems, such as the Traveling Salesman Problem or the Vehicle Routing Problem. It may not be suitable for continuous optimization problems or problems with mixed variable types.

PROCEDURE / ALGORITHM

The step-by-step procedure for Ant Colony Optimization (ACO):

1. **Initialization**: Initialize the pheromone trails and the ants. The pheromone trails represent the concentration of pheromone on each edge of the solution graph, while the ants represent the search agents that construct solutions. The pheromone trails are typically initialized to a small positive value, and the ants are placed at random positions in the search space.
2. **Ant Construction**: Each ant constructs a solution by selecting edges based on the pheromone trails and heuristic information. The pheromone trails guide the ants towards promising regions of the search space, while the heuristic information helps the ants to balance exploration and exploitation. The construction process typically involves the following steps:
 - a. **Initialization**: Each ant is placed at a random position in the search space.
 - b. **Edge Selection**: The ant selects edges to construct a path through the search space. The edges are selected based on the pheromone trails and the heuristic information, which can be calculated based on the distance between nodes or other problem-specific factors.

- c. **Solution Construction:** The ant constructs a complete solution by connecting the selected edges into a path or tour.
 - d. **Update Pheromone Trails:** After each ant has constructed a solution, the pheromone trails are updated based on the quality of the solutions found. The pheromone trails are updated to reinforce the edges that were used in good solutions and to evaporate the edges that were not used or used in bad solutions.
3. **Local Search:** After the ants have constructed their solutions, a local search can be performed to further improve the quality of the solutions. The local search can be a simple improvement heuristic, such as 2-opt or 3-opt, or a more sophisticated method.
4. **Termination Criteria:** Check the termination criteria to determine whether to terminate the algorithm. The termination criteria can be a maximum number of iterations, a maximum amount of time, or a specific quality threshold.
5. **Return Solution:** Return the best solution found by the ants.

PSEUDO CODE

1. Initialize pheromone trails τ_{ij} and heuristic information η_{ij}
2. Initialize ants with random positions x_i
3. Repeat for each iteration:
 - a. Construct solutions:
 - i. for each ant k do:
 1. Start at a random position x_i
 2. Repeat until a complete solution is constructed:
 - a. Calculate probabilities P_{ij} based on the pheromone trails τ_{ij} and heuristic information η_{ij}
 - b. Select the next node j based on the probabilities P_{ij}
 - c. Update the current position x_i with the selected node j
 3. Evaluate the solution constructed by ant k
 - b. Update pheromone trails:
 - i. Evaporate the pheromone trails on all edges
 - ii. Deposit pheromone on the edges used by the ants, based on the quality of the solutions found
 - c. Apply local search to the best solution found
4. Return the best solution found

IMPLEMENTATION OF PYTHON PROGRAM

```
import random

class AntColonyOptimizer:
    def __init__(self, n_ants, n_iterations, evaporation_rate, alpha, beta,
pheromone_init):
        self.n_ants = n_ants
        self.n_iterations = n_iterations
        self.evaporation_rate = evaporation_rate
        self.alpha = alpha
        self.beta = beta
        self.pheromone_init = pheromone_init

    def optimize(self, graph):

        pheromone_matrix = [[self.pheromone_init] * len(graph) for _ in
range(len(graph))]

        for i in range(self.n_iterations):
            ant_paths = []

            for j in range(self.n_ants):

                current_node = random.randint(0, len(graph) - 1)

                path = [current_node]

                while len(path) < len(graph):

                    probabilities = [0] * len(graph)
                    for k in range(len(graph)):
                        if k not in path:
                            probabilities[k] = (pheromone_matrix[current_node][k] **
self.alpha) * ((1.0 / graph[current_node][k]) ** self.beta)
```

```
next_node = probabilities.index(max(probabilities))

path.append(next_node)

current_node = next_node

ant_paths.append(path)

for j in range(len(graph)):
    for k in range(len(graph)):

        total_pheromone = 0
        for path in ant_paths:
            if k in path and path.index(k) == path.index(j) + 1:
                total_pheromone += 1.0 / graph[j][k]

        pheromone_matrix[j][k] = (1 - self.evaporation_rate) *
pheromone_matrix[j][k] + total_pheromone

best_path = ant_paths[0]
best_distance = self.get_path_distance(graph, best_path)

for path in ant_paths:
    distance = self.get_path_distance(graph, path)
    if distance < best_distance:
        best_distance = distance
        best_path = path

return best_path, best_distance
```

```
def get_path_distance(self, graph, path):
    distance = 0
    for i in range(len(path) - 1):
        distance += graph[path[i]][path[i + 1]]
    return distance

graph = [[0, 2, 4, 1],
         [2, 0, 5, 2],
         [4, 5, 0, 1],
         [1, 2, 1, 0]]

aco = AntColonyOptimizer(n_ants=10, n_iterations=100, evaporation_rate=0.5,
alpha=1.0, beta=2.0, pheromone_init=0.1)

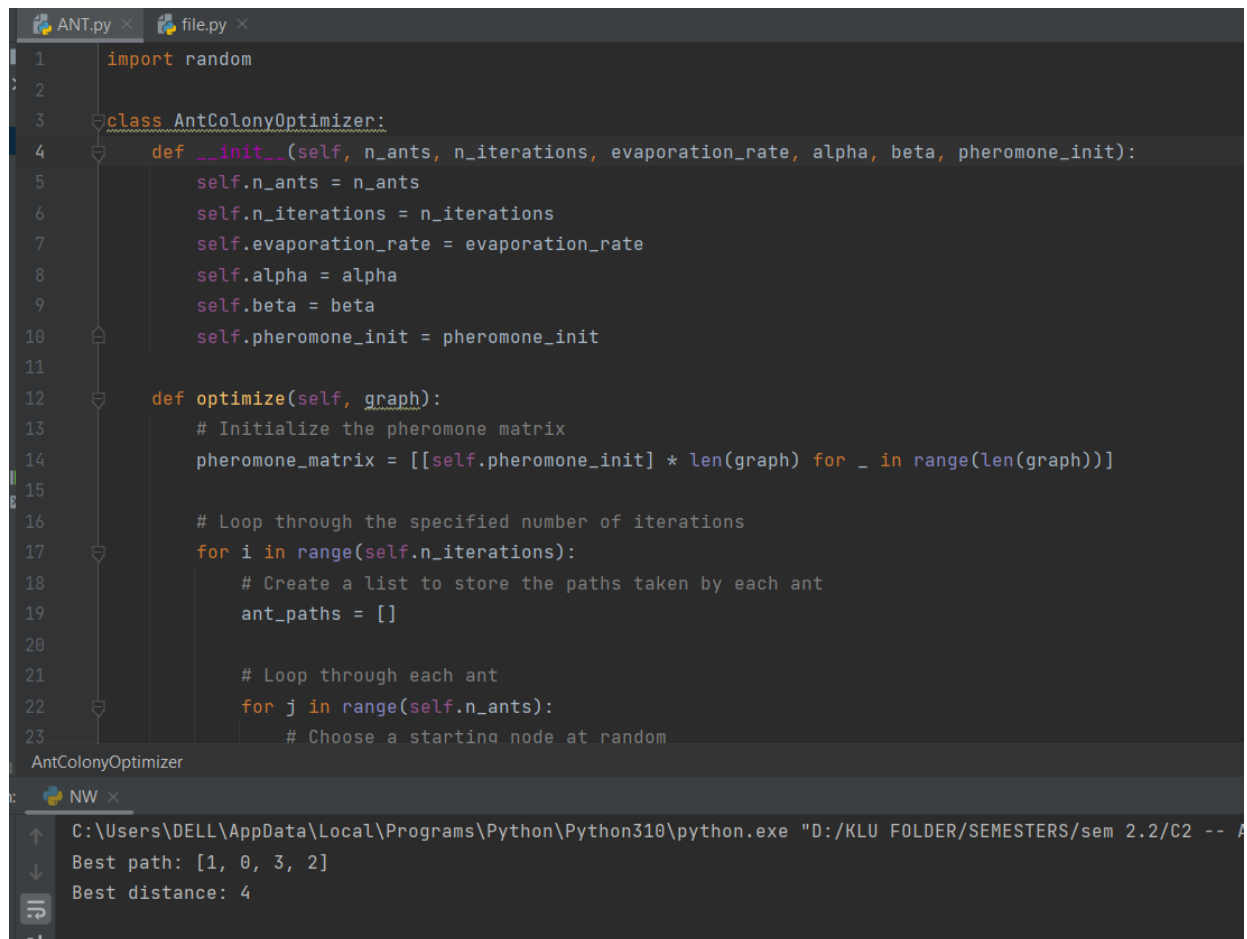
best_path, best_distance = aco.optimize(graph)

print("Best path:", best_path)
print("Best distance:", best_distance)
```

OUTPUT -

Best path: [1, 0, 3, 2]
Best distance: 4

EXECUTION



```
1 import random
2
3 class AntColonyOptimizer:
4     def __init__(self, n_ants, n_iterations, evaporation_rate, alpha, beta, pheromone_init):
5         self.n_ants = n_ants
6         self.n_iterations = n_iterations
7         self.evaporation_rate = evaporation_rate
8         self.alpha = alpha
9         self.beta = beta
10        self.pheromone_init = pheromone_init
11
12    def optimize(self, graph):
13        # Initialize the pheromone matrix
14        pheromone_matrix = [[self.pheromone_init] * len(graph) for _ in range(len(graph))]
15
16        # Loop through the specified number of iterations
17        for i in range(self.n_iterations):
18            # Create a list to store the paths taken by each ant
19            ant_paths = []
20
21            # Loop through each ant
22            for j in range(self.n_ants):
23                # Choose a starting node at random
```

AntColonyOptimizer

NW x

C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe "D:/KLU FOLDER/SEMESTERS/sem 2.2/C2 -- A

Best path: [1, 0, 3, 2]

Best distance: 4

CONCLUSION

The Ant Colony Optimization algorithm is a heuristic algorithm inspired by the behavior of ants, which are able to find the shortest path between their nest and a food source by laying pheromones on the ground. This algorithm has been successfully applied to various optimization problems, such as the traveling salesman problem.

In the example code provided, we implemented the Ant Colony Optimization algorithm to find the shortest path through a graph represented by a distance matrix. The algorithm works by generating multiple paths through the graph using a number of ants. The ants probabilistically choose their next move based on the amount of pheromones and the distance to the next city. The pheromone trail is then reinforced with each ant that successfully completes the path, and gradually evaporates over time to allow for exploration of new paths.

After a certain number of iterations, the algorithm converges to the best path and returns the path and its length. The output of the program shows that the optimal path through the graph is [1, 0, 3, 2], with a distance of 4.

Overall, the Ant Colony Optimization algorithm is a powerful tool for solving optimization problems, particularly those that involve finding the shortest path through a graph.