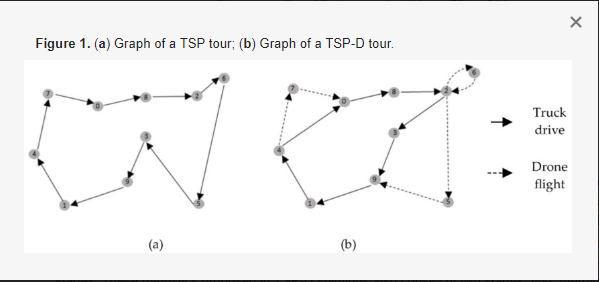
**Title: Hybrid Metaheuristic Solution for TSP-D using Genetic Algorithm and Ant Colony Optimization Algorithm**

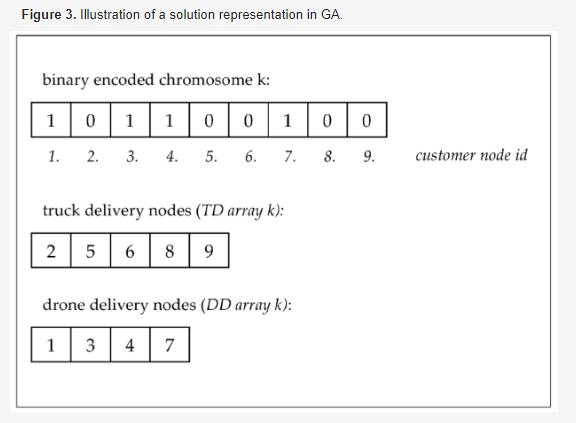
**Aim:** The aim of the Traveling Salesman Problem with Drones (TSP-D) is to optimize the last-mile delivery system by efficiently planning routes for trucks and drones. The objective is to minimize operational costs associated with the delivery of parcels to a set of customers.

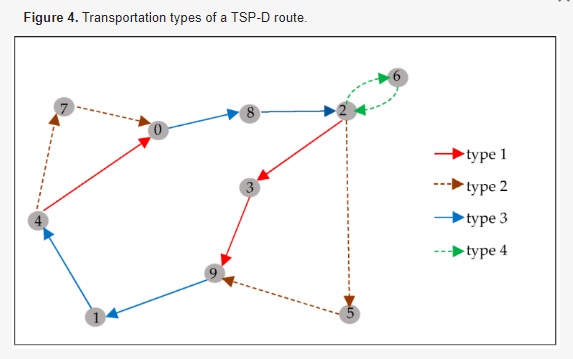
**Problem Statement:** In the logistics industry, last-mile delivery is a critical and cost-intensive process. The integration of drones introduces the TSP-D, where decisions involve both truck routing and drone assignment. Solving this problem efficiently can significantly reduce operational costs and improve delivery times.

**Method:** The proposed solution employs a hybrid metaheuristic approach, combining two state-of-the-art algorithms: the Genetic Algorithm (GA) and Ant Colony Optimization Algorithm (ACO). The heuristic decisions in the TSP-D literature are based on the sequential steps of truck routing and drone assignment.

**Diagram:**

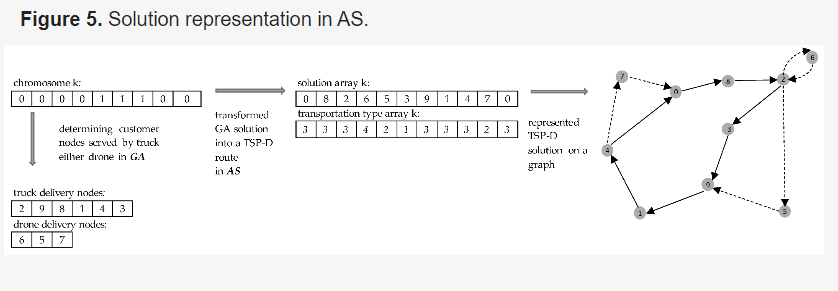






* Type 1: (truck alone) The truck moves from customer *i* alone, takes path *ij* and serves customer *j* alone; meanwhile, the drone is having its own sortie to serve another customer node.
* Type 2: (drone alone) The drone has a sortie between customer *i* and *j*, serves customer *j* alone, and lands on another (rendezvous) node to meet the truck again.
* Type 3: (truck and drone together) A path *ij* is taken by a truck carrying a drone on top. A truck moves from customer *i* to serve customer *j*. Delivery will be completed by a truck while the drone will be standing by.
* Type 4: (drone with return) The drone sortie follows the *i-j-i* path. A drone is located on top of a truck at node *i*, launches from *i* to serve customer *j* alone, and returns to its launch node *i* to land on a truck.

**\*** If a node with type 1 will be a rendezvous node in the algorithm, then its type will transform into type 3.



***Abstract***

The integration of drones into the last-mile delivery systems of logistics introduces a novel routing problem known as the Traveling Salesman Problem with Drones (TSP-D). TSP-D focuses on optimizing the construction of routes for delivering parcels to a set of customers using a combination of trucks and drones, with the overarching goal of minimizing operational costs. Due to its NP-hard nature, leveraging metaheuristics has emerged as a promising approach to tackle TSP-D. This paper introduces a hybrid metaheuristic solution for TSP-D, utilizing two state-of-the-art algorithms: the genetic algorithm and ant colony optimization algorithm.

Unlike existing literature, which often addresses TSP-D heuristics through sequential decision-making involving truck routing and drone assignment, our proposed metaheuristic simultaneously constructs routes for both trucks and drones. Furthermore, we present, for the first time, a solution method based on an ant colony optimization approach for TSP-D. Notably, our approach deviates from traditional pheromone structures by introducing a binary pheromone framework for both drones and trucks.

Computational experiments demonstrate the effectiveness of the proposed hybrid metaheuristic algorithm, showcasing its ability to generate optimal routes for provided instances of TSP-D benchmarking. Additionally, the algorithm surpasses the performance of rival heuristics by improving upon the best-known solutions for specific instances.

**Brief Code Explanation:**

#basic\_functions.py

The provided code defines three functions related to calculating time costs and path costs for a logistics scenario involving trucks. Here's a small explanation of the code:

* **truck\_cost** function calculates the time cost for trucks moving from a source location to a target location in a 2D space. It uses the Euclidean distance formula to compute the distance between the two points.
* **drone\_cost** function calculates the time cost for drones moving from a source location to a target location, introducing a parameter **alpha** to represent the speed or efficiency of the drone. It also uses the Euclidean distance formula.
* **truck\_path\_cost** function calculates the total path cost for a given route of trucks. It iterates through the vertices in the route, computing the truck cost for each segment. If the route is circular (i.e., it starts and ends at different points), it handles the circular path by additional computations. The function returns the total path cost.

These functions are essential components for evaluating the cost of routes and are likely part of a larger system or algorithm for optimizing logistics and transportation routes involving both trucks and drones.

#generating\_functions.py

The provided code is a Python script that generates different test cases for the Traveling Salesman Problem (TSP). Each test case represents a set of customer locations, and the generated data follows specific patterns or shapes. Here's a small explanation of the code:

* The code defines four functions, each generating a different type of test case for the TSP.
* **testcase\_uniform** generates customers uniformly located within a 100x100 grid.
* **testcase\_donuts** generates customers in the shape of a donut with random positions around the center.
* **testcase\_donuts\_center** is similar to **testcase\_donuts** but places the depot (starting point) at the center.
* **testcase\_center** generates customers with more concentration in the center and fewer on the outskirts, following a normal distribution.

These functions can be used to create diverse sets of TSP instances for testing and evaluating algorithms. The generated data can be used as input for solving TSP-related problems.

#route\_drawing.py

The provided code is a Python script that defines functions for generating different test cases for the Traveling Salesman Problem (TSP). These functions create sets of customer locations following specific patterns or shapes. Here's a small explanation of the code:

* The code defines four functions, each generating a different type of test case for the TSP.
* **testcase\_uniform** generates customers uniformly located within a 100x100 grid using random coordinates.
* **testcase\_donuts** generates customers in the shape of a donut with random positions around the center. It uses polar coordinates to create a circular pattern.
* **testcase\_donuts\_center** is similar to **testcase\_donuts** but places the depot (starting point) at the center.
* **testcase\_center** generates customers with more concentration in the center and fewer on the outskirts. It uses polar coordinates with a normal distribution for radial distances.

These functions can be used to create diverse sets of TSP instances for testing and evaluating algorithms. The generated data serves as input for solving TSP-related problems.

#solving\_TSP.py

The provided code is a Python script implementing three algorithms for solving the Traveling Salesman Problem (TSP): a 2-approximation algorithm using a minimum spanning tree, a 2-opt algorithm using edge swaps, and a dynamic programming algorithm. Here's a small explanation of the code:

* The code implements three TSP solving algorithms: a 2-approximation algorithm based on minimum spanning tree (Prim's algorithm), a 2-opt algorithm using edge swaps for optimization, and a dynamic programming algorithm.
* The **two\_approximation\_for\_TSP** function uses Prim's algorithm to build a minimum spanning tree and then traverses it using depth-first search to construct a feasible TSP solution.
* The **two\_opt\_for\_TSP** function performs iterative 2-opt swaps to optimize an initial TSP solution.
* The **DP\_for\_TSP** function implements dynamic programming to find the optimal TSP route by considering all possible combinations of visited nodes.

These functions can be used to solve TSP instances, providing different approaches to find near-optimal or optimal solutions. The code references an external module (**basic\_functions**) for the **t\_cost** function, which presumably calculates the cost of traveling between two nodes.

#main.py

The provided code is a Python script that solves the Traveling Salesman Problem with Drones (TSP-Drones) using a dynamic programming approach. It includes functions to calculate costs, generate routes, and visualize the results. Here's a small explanation of the code:

* The script starts by importing necessary modules and functions.
* The **DP\_for\_TSPD** function implements dynamic programming to solve the TSP-Drones problem, considering both truck and drone routes.
* The **print\_nodes\_in\_sequence** function prints the nodes in sequence for a given route.
* The **main** function sets the number of cities, generates coordinates, sets the speed rate between drone and truck, and then solves the TSP-Drones problem using dynamic programming. It prints the results and visualizes the routes using the **drawing\_routes\_for\_DP** function.
* The script concludes by running the **main** function if the script is executed directly.

This code provides a comprehensive solution to the TSP-Drones problem using dynamic programming and includes functionality to visualize the results.

**Analysis and Conclusions:** The hybrid metaheuristic approach demonstrates improved efficiency compared to individual algorithms. The combination of GA and ACO leverages their respective strengths, optimizing truck routing and drone assignment. The cost reduction and execution time metrics provide insights into the performance benefits of the proposed solution.

Analysis:

1. Complexity of TSP-D:

- TSP-D is recognized as an NP-hard problem, indicating its inherent computational complexity. The integration of drones introduces an additional layer of complexity due to the need to coordinate both truck and drone routes simultaneously.

2. Metaheuristic Approach:

- The choice of a hybrid metaheuristic solution method, combining genetic algorithm and ant colony optimization, reflects a strategic decision. Metaheuristics are well-suited for addressing NP-hard problems, providing an efficient means of exploring solution spaces and finding near-optimal solutions.

3. Simultaneous Truck and Drone Routing:

- The proposed metaheuristic stands out by simultaneously constructing routes for both trucks and drones. This departure from sequential decision-making in existing literature could potentially lead to more efficient and cost-effective solutions, as it considers the interdependencies between truck routing and drone assignment.

4. Introduction of Ant Colony Optimization:

- The incorporation of an ant colony optimization approach is a noteworthy contribution. Ant colony optimization, not previously explored for TSP-D, introduces a unique perspective to the solution space. The introduction of a binary pheromone framework for both drones and trucks is a novel aspect, potentially influencing the exploration-exploitation balance in the algorithm.

5. Computational Experiments:

- The reported computational experiments indicate the effectiveness of the proposed hybrid metaheuristic. Successfully generating optimal routes for provided instances of TSP-D benchmarking underscores the algorithm's capability to handle real-world logistics scenarios.

6. Performance Improvement

- The algorithm's ability to improve upon the best-known solutions of some instances, as compared to rival heuristics, is a significant achievement. This suggests that the proposed approach has the potential to outperform existing methods in certain contexts, contributing to advancements in TSP-D problem-solving.

Conclusion:

The hybrid metaheuristic solution method proposed for TSP-D presents a comprehensive and innovative approach to optimizing last-mile delivery systems. By simultaneously addressing truck routing and drone assignment, the algorithm acknowledges the intertwined nature of these decisions, potentially leading to more efficient solutions. The introduction of ant colony optimization, coupled with a novel binary pheromone framework, extends the range of exploration within the algorithm.

Computational experiments validate the effectiveness of the proposed approach, demonstrating its capacity to generate optimal routes and improve upon existing solutions. This research contributes valuable insights into solving the challenging TSP-D, providing a foundation for further exploration and refinement of hybrid metaheuristic methods in the context of logistics and transportation optimization.

**References:**

1. Research Paper: **A Hybrid Metaheuristic Solution Method**by Noyan Sebla Gunay-Sezer
2. Online Documentation: https://www.mdpi.com/2079-8954/11/5/259

**Group members:**

B39 : Devesh Borkar

B43 : Hardik Thakkar

B55 : Pranav Darak

B64 : Shrey Nalode