## Problem Description

The **Decision Support System (DSS)** for ICAERUS is designed to optimize UAV logistics using a **Capacitated Vehicle Routing Problem (CVRP)** solver. The system includes **compiled execution (model.exe)**, improved **route storage and retrieval**, and a **refactored class-based structure** (dss_class.php).

This implementation extends the traditional **Capacitated Vehicle Routing Problem (CVRP)** by integrating **Vehicle Routing with Pickups and Deliveries (VRPD)** into a unified model. Unlike standard CVRP solvers, which focus only on delivery constraints, this approach dynamically incorporates both delivery and pickup demands within the same route, ensuring optimized fleet utilization. By leveraging **Google OR-Tools**, this model allows for complex constraints such as:

- **Maximum vehicle range**
- **Structured demand requests**
- **Cargo dimension constraints** (max cargo length, width, and height)
- **Geospatial mapping support** for UAV operations

This hybrid method is particularly useful for real-world drone operations, where simultaneous deliveries and pickups improve efficiency and resource utilization.

## Solution Summary

### Technologies & Tools Used

- **Google OR-Tools** for route optimization
- **PHP & MySQL** for backend data storage
- **Leaflet.js** for interactive route visualization
- **Compiled Execution (model.exe)** for standalone UAV route processing
- **REST API** for backend integration

### Techniques Used

- Constraint Programming
- Distance Matrix Calculation
- Delivery and Pickup Constraints Handling

### Methods

1. **Data Processing:**
    - Reads vehicle attributes, including capacity, range, and cargo dimensions.
    - Parses structured demand requests for deliveries and pickups.
    - Stores geospatial coordinates for accurate route planning.

2. **Optimization & Constraints Handling:**
    - Uses **OR-Tools** to optimize UAV route planning.

- Implements **vehicle-specific constraints** (capacity, max distance, cargo size limitations).
- Ensures vehicles return to the depot after completing assigned routes.

3. **Execution Workflow:**
    - UAVs and orders are filtered dynamically before optimization.
    - The compiled executable (model.exe) processes cvrp.json and outputs cvrp_solution.json.
    - Routes are stored and retrieved via a **REST API** for seamless integration

4. **Visualization & Output Processing:**
    - **Leaflet.js** is used to display optimized routes interactively.
    - A **MySQL database** stores past route solutions for future analysis.

## Code Analysis

1. **Problem Definition and Data Setup:**
    - The create_data_model function defines the problem data, including the distance matrix, structured demand requests (delivery & pickup), vehicle capacities, depot index, maximum distance, and optional geospatial coordinates.

2. **Callback Functions:**
    - create_distance_callback: Defines a callback function that converts location indices to node values and retrieves corresponding distances from the indexed distance matrix.
    - create_delivery_and_pickup_callbacks: Creates separate callbacks for delivery and pickup demands at each location.

3. **Routing Model Setup:**
    - The routing model is initialized using the pywrapcp.RoutingModel class, with the index manager and number of vehicles specified.
    - Cost and distance constraints are defined using the distance callback and added to the model.

4. **Capacity Constraints:**
    - Capacity constraints are added for both deliveries and pickups using the AddDimensionWithVehicleCapacity method.
    - Separate capacity dimensions are added for deliveries and pickups, ensuring that each vehicle respects its capacity limit.

5. **Solution Search:**

    - Search parameters are defined using pywrapcp.DefaultRoutingSearchParameters, specifying the first solution strategy.
    - The problem is solved using the SolveWithParameters method.

6. **Solution Analysis and Output Processing:**
    - The print_solution function analyzes and prints the solution routes for each vehicle.
    - It iterates over each vehicle's route, calculating the total distance traveled, delivery and pickup loads at each step, and prints the route details accordingly.
    - The get_solution function processes the solution data into a list of dictionaries, providing a structured representation of the solution routes.
    - This structured data can be easily used for further analysis, visualization, or integration with other systems.

## Deliverable

- An optimized UAV routing plan ensuring:
    - Vehicles respect delivery and pickup constraints.
    - Total distance traveled is minimized.
    - Routes are stored for retrieval and visualization.

## Advantages of the DSS System

- **Standalone Execution:** Eliminates the need for a Python runtime using model.exe.
- **Scalability:** Supports real-time route adjustments and multiple UAV hubs.
- **Enhanced Efficiency:** Integrates both delivery and pickup logistics for optimized fleet usage.
- **Geospatial Accuracy:** Uses precise coordinate-based routing for UAV operations.

## Areas for Future Improvement

- Implement **live UAV tracking** integration.
- Add **multi-depot optimization** for improved logistics efficiency.
- Enhance **AI-driven demand forecasting** for better resource allocation.
- Develop an **advanced analytics dashboard** to visualize performance metrics.

## Conclusion

The ICAERUS DSS system provides a robust and scalable solution for UAV logistics, leveraging **OR-Tools** and best practices in **constraint programming**. The combination of **compiled execution, structured data storage, and interactive visualization** ensures efficiency, scalability, and seamless integration into real-world drone operations.