

School Bus Routing Optimization

1. Project Overview

This project optimizes school bus routes to reduce transportation costs and improve student safety. The web app is built using **Streamlit**, and it leverages **Google's OR-Tools** for route optimization and **Optuna** for hyperparameter tuning.

The optimization is based on minimizing total travel distance while respecting constraints like the number of buses and student capacity per bus.

2. Technologies Used

- **Streamlit** (for Web UI)
 - **OR-Tools** (for solving Vehicle Routing Problem)
 - **Optuna** (for parameter optimization)
 - **Python, NumPy, FPDF**
-

3. User Inputs

Users control the optimization using these adjustable settings in the sidebar:

- **Min Buses:** Minimum number of buses to consider for optimization
 - **Max Buses:** Maximum number of buses to consider
 - **Min Capacity:** Minimum bus capacity (e.g., 40 students)
 - **Max Capacity:** Maximum bus capacity (e.g., 60 students)
 - **Optimization Trials:** Number of Optuna iterations to try different combinations
-

4. Sample Locations (Chennai, India)

The app uses sample coordinates for major areas in Chennai to simulate pickup points:

- T. Nagar (Depot)
- Adyar
- Anna Nagar
- Guindy
- Mylapore
- Kilpauk

5. Optimization Logic

- A distance matrix is created using the coordinates of all locations.
 - **OR-Tools** solves the **Vehicle Routing Problem (VRP)** with:
 - A depot (start and end location for all buses)
 - Student demand per stop
 - Capacity limits for each bus
 - **Optuna** is used to try different bus counts and capacities and select the best performing configuration.
-

6. Optimization Techniques Used

Google OR-Tools:

- Solves the VRP using **local search** methods:
 - **Tabu Search**
 - **Guided Local Search**
- Ensures every location is visited once, without exceeding bus capacity.

Optuna:

- Uses **Bayesian Optimization** via the **Tree-structured Parzen Estimator (TPE)**.
 - Automatically tries different combinations (bus count, capacity).
 - Evaluates the **total distance** and finds the configuration that minimizes it.
-

7. Detailed Optimization Workflow

1. User sets slider ranges for buses and capacities (e.g., 2–5 buses, 40–60 capacity).
 2. Optuna runs multiple trials (e.g., 20), selecting random values within these limits.
 3. For each trial:
 - A set number of buses and capacity are passed to OR-Tools.
 - OR-Tools generates optimized routes using local search algorithms.
 4. Total distance is calculated for the trial.
 5. The best result (minimal distance) is displayed to the user.
-

8. Output

- 🚍 Best total distance to transport all students
- 🚍 Optimized number of buses and capacity
- 🌈 Route map showing all locations

The image shows a dual-screen setup. The top screen is a code editor (VS Code) displaying Python code for solving bus routing. The bottom screen is a web application interface titled "School Bus Route Optimization".

Code Editor (VS Code):

- File Explorer:** Shows files like `app.py`, `.venv`, and `OT MI...`.
- Terminal:** Shows command-line output related to the optimization process.
- Output:** Shows logs indicating a new study was created and a trial finished with a value of 20446.0.

```
40 def solve_bus_routing(num_buses, bus_capacity):
    search_parameters.local_search_metaheuristic = (
        routing_enums_pb2.LocalSearchMetaheuristic.GUIDED_LOCAL_SEARCH)
    search_parameters.time_limit.seconds = 5
    solution = routing.SolveWithParameters(search_parameters)

    total_distance = 0
    if solution:
        for vehicle_id in range(num_buses):
            index = routing.Start(vehicle_id)
            route_distance = 0
            while not routing.IsEnd(index):
                previous_index = index
                index = solution.Value(routing.NextVar(index))
                route_distance += routing.GetArcCostForVehicle(previous_index, index, vehicle_id)
            total_distance += route_distance
```

Web Application (School Bus Route Optimization):

- Settings:** Includes sliders for "Min Buses" (set to 2), "Max Buses" (set to 5), "Min Capacity" (set to 40), "Max Capacity" (set to 60), and "Optimization Trials" (set to 5).
- Run Optimization:** A button to start the optimization process.
- Results:** Displays the "Best Total Distance" as "20446.0 meters".
- Code Snippet:** Shows the JSON configuration used: `{"num_buses": 3, "bus_capacity": 50}`.
- Map:** A map of Chennai, India, showing the optimized bus routes. Labeled areas include AIR, NAGAR, SHENDY NAGAR, AMINJIKARAI, CHETPUT, EGORE, ISLAND GROUNDS, RAM, NGO COLONY, NUNGAMBAKKAM, MAHALINGAPURAM, TRIPOLICANE, MATTANKUPPAM, SALIGRAMAM, Kaveri Rangan Colony, DAMBAAKKAM, and KODAMBAAKKAM.