Real Time Route Analysis Project Report

Buğra Tuncer
Advanced Practical Course - Run-Time Data Visualization for Business Processes
ge27lol@mytum.de

I. Model and Server Setup Phase

A. Running the Server

Detailed instructions for running the server can be found in the repository's *README* file. In short, after cloning or directly accessing the repository, the user should start the program using the command: **python3 app.py**

B. Setting Parameters in CPEE

While the server is running, the test set should be loaded into CPEE. The user should configure the following parameters in the **Data Elements** section to start the model without an issue:

- **waypoints**: The starting and destination locations.
- ***instance_id***: Must match the instance ID provided by CPEE.

II. Initialization Phase

A. Provide Instance ID / Initialize-Get Car Object

ID-Endpoint: a10, *update_instance_id*

This endpoint initializes the car object with the given waypoints and instance ID. If a car object with the specified *instance_id* already exists, it returns that object; otherwise, it creates a new one. The response car object stored in *car* object in the **Data Elements** section.

B. Retrieve Coordinates via TomTom Geocoding API

ID-Endpoint: a29, *get_city_coordinates*

This endpoint returns the latitude and longitude of the given waypoints and stores them in the *coordinates* list in the **Data Elements** section.

C. Plan/Get Route

ID-Endpoint: a6, *get_route_data*

Based on the *coordinates* list, the server creates a route using the **Geoapify Route API**. The initial route is saved to: **initial_route_[instance_id].json**

Additionally, a background thread starts and updates the car's location based on elapsed time. This file is also linked to the visualization dashboard to display the route on the map.

To calculate the car's current location, the system uses Coordinates and Route objects from the file. The Route object contains information about how long it takes to complete a specific section of the coordinates. Based on the elapsed time and the route's time value, the car's new location is determined using data from the Coordinates object.

III. Right Side of The Parallel Execution - Data Retrieval Phase

A. Get Traffic Data

ID-Endpoint: a9, *get_traffic_data*

This endpoint fetches live traffic updates using the **TomTom Traffic Flow API**. The response is stored in the **Data Elements** section under *trafficResponse*.

- Data probes capture the response via CPEE POST requests and store it in: **historical_data_[instance_id].json**
- Server-Sent Events (SSE) update the visualization dashboard. (Traffic Data, SRF Contribution Analysis Radar Chart, TomTom Live Speed)
- This loop runs until the car reaches the *"arrived"* state and executes within the *data_retrieval_time* interval.

B. Get Weather Data

ID-Endpoint: a18, *get_weather_data*

This endpoint fetches weather updates using the **Weather API**. The response is stored in the **Data Elements** section under *weatherResponse*.

- Data probes capture the response via CPEE POST requests and store it in: **historical_data_[instance_id].json**
- Server-Sent Events (SSE) update the visualization dashboard. (Weather Data, SRF Contribution Analysis Radar Chart)
- This loop runs until the car reaches the *"arrived"* state and executes within the *data_retrieval_time* interval.

C. Get Latest Car State

ID-Endpoint: a5, *get_car_state*

The response is stored in the *car* element in the **Data Elements** section. This endpoint provides the latest car position, calculated by background server operations.

- Since the car's position updates more frequently than weather/traffic data, it executes within the *car_retrieval_time* interval.
- Typically, *car_retrieval_time* is half of *data_retrieval_time*.

IV. Left Side of The Parallel Execution - SRF Factor, Car Modificiation Phase

Since left side of the parallel requires information about the weatherResponse, trafficResponse and updated car. For the initial loop there is a 10 seconds delay for these operations to run successfully.

A. Speed Reduction Factor (SRF) Calculation

ID: Various Scripts(a32, a3, a21, a31, a30, a20, a16)

Multiple scripts calculate a **Speed Reduction Factor (SRF)**, referred to as *speed_calculation_threshold* in the **Data Elements** section. The SRF is influenced by:

- **Confidence Value:** From *trafficResponse* (0.0 to 1.0; higher = more reliable).
- **Traffic Density:** Derived from comparing *free_flow_speeds* and *live_speeds* (larger difference = possible congestion).
- **Incidents/Road Closures:** Significantly impact SRF if present in *trafficResponse*.
- **Visibility Value:** From *weatherResponse* (0.0–10 km).
- **Weather Conditions:** (e.g., "Blizzard," "Clear," "Fog") affect visibility and SRF.

B. Move Car with Updated Speed

ID-Endpoint: a7, *update_car_speed*

This endpoint updates the car's speed using the calculated SRF factor and *instance_id*. The response is stored in the *car* element.

- Data probes capture the response via CPEE POST requests and store it in: **historical_data_[instance_id].json**
- Server-Sent Events (SSE) update the visualization dashboard (Car Data, Calculated Car Speed Data, SRF Factor Information, SRF Contribution Analysis Radar Chart, Active Route Display - Current Position of the Car in the Route).

C. Car Light Control

- **Turn On/Off Car Fog Lights**

  ID-Endpoint: a26-a15, *update_car_fog_lights*

  The fog lights are controlled based on weather conditions stated in the *weather_control* list in **Data Elements** section.
- **Turn On/Off Car Lights**

  ID-Endpoint: a12-a23, *update_car_lights*

  The headlights are controlled based on the current time.

Since Calculated Car Speed is an important factor to visualize on the dashboard, these services do not directly connect to the data streams. After they complete their modifications on the car, the next cycle of the Update Speed data stream will push the updated speed information to the visualization dashboard.

V. Aspects Comparison with Project Proposal

Component	Proposal Plan	Implementation
Process Scope	Single-city traffic analysis	Route-based analysis between two locations
Data Sources	• WeatherAPI • TomTom Vector Flow Tiles API	• WeatherAPI (retained) • TomTom Flow Data API (simplified route-specific data) • Geoapify Route API (new addition)
Process Model	• CPEE operations without sensor integration	• Parallel execution model: – Left side: SRF calculation using sensor values (weather/traffic) – Right side: Data retrieval
Information Panel	• Instance Information • Current Temperature-Weather Conditions • Current Traffic Level Vectoral Tile Map - Heatmap	• Weather Data • Traffic Data • Car Data • Speed Reduction Factor Information
Graphs	6 graphs: • Real-Time Traffic Flow • Real-Time Temperature Graph • Weather Conditions vs. Travel Time Graph • Weather Conditions vs. Average Traffic Speed Graph • Temperature vs. Traffic Level Graph • Weather Conditions vs. Incidents Graph	Optimized 4 dynamic views: • **Speed Reduction Factor Contribution Analysis Radar Chart**: Real-time SRF components – Weather conditions – Traffic density – Visibility – Road Closures - Incidents – Confidence • **TomTom Live Speed vs Calculated Car Speed**: – Calculated Car Speed – TomTom Live Speed • **Calculated Car Speed vs Speed Limit**: – Calculated Car speed – Speed Limit – Speed Reduction Factor • **Active Route Display**: – Historical Route – Current Position of the Car

VI. Known Issues

- **Process Stuck in Wait State (Powernap)**: The process may appear as ”running” but become unresponsive during wait operations. To resolve:
  - Manually remove the ”execute from here” sections
  - Restart the process - it will automatically resume from the last active state
- **Delays under heavy CPEE load**: During high server load, CPEE Post Requests may be delayed even when data streams return responses. Since the visualization requires synchronized Car, Weather, and Traffic data,

instance file creation can take over a minute. If the instance file doesn't appear after using reload button, please wait 2-3 minutes before checking again.

VII. Project Related Links

- Github Repository
- Lehre Server Code Directory : /srv/gruppe/students/ge27lol/public_html/Route-Analysis-Project
- Model Directory
- Visualization Website