Opti'tour Glossary

Jason LAVAL

Abstract—This document explanes the basic vocabulary and defines the key terms for this project.

Keywords—*Actors,Use Cases, Buisiness Rules, State Definitions*

Contents

1 Actors **1**

1.1 Dispatcher 1

1.2 Courier 1

2 Use Cases **1**

3 Business Rules **1**

4 State Definitions **1**

5 Overview of the Model **1**

1. Actors

1.1. Dispatcher

The dispatcher is responsible for managing the delivery operations. They load the city map, adjust the number of couriers, record delivery requests, and oversee all tours. When a tour cannot be computed for a courier, the dispatcher decides whether to assign another courier or to reject the request.

1.2. Courier

The courier is a bicycle delivery worker who follows the assigned route. They start from the warehouse at 8:00 a.m., complete all pickups and deliveries in order, and return to the warehouse at the end of the tour.

2. Use Cases

1. Load City Map (XML)
The dispatcher loads an XML file describing the city layout, including all intersections, streets, and the warehouse location.

2. Display Map
The system shows the loaded city map on screen. It becomes the visual base for displaying future tours.

3. Set Number of Couriers
The dispatcher defines how many couriers are available for deliveries. By default, the system starts with one courier.

4. Add a Courier
The dispatcher can add a new courier.

5. Load delivery Request
The dispatcher receives a new delivery request from a customer (by phone, message, or email) in XML format. The request describes where to pick up and where to deliver a package. He can load them to compute best tour in the future.

6. Compute Best Tour
The system automatically calculates an optimised possible tour for the chosen courier. The goal is to return to the warehouse as early as possible after visiting all pickups and deliveries.

7. Display Tour on Map
The system displays the calculated route. For each stop the arrival time time are shown.

8. Ask to Select Another Courier
If no valid tour can be built for the selected courier, the dispatcher is invited to choose another one.

9. Reject Delivery Request
If no courier can handle the delivery, the dispatcher must reject the request.

10. Save Current Tours to File
The dispatcher saves the current set of tours and deliveries for later use.

11. Restore Tours from File
The dispatcher reloads previously saved tours. When tours are restored, the map and all routes are displayed again.

12. Execute Assigned Tour
The courier follows the planned route, performs all pickups and deliveries in the given order, and returns to the warehouse once the route is completed.

3. Business Rules

1. All tours start at **8:00 a.m.** from the warehouse.
2. Couriers travel at a constant speed of **15 km/h**.
3. The system always seeks to **optimise the return time** to the warehouse.
4. For each request, the pickup location must be visited**before** its corresponding delivery location.
5. All tours begin and end at the **same warehouse**.
6. Each courier's tour is displayed on the map with all stops, addresses and arrival times.
7. If no valid route can be found for one courier, the system proposes another courier; if no solution exists, the request must be rejected.
8. Tours can be saved and restored at any time to continue planning later.

4. State Definitions

Feasible Tour
A tour that satisfies all business rules: pickups before deliveries, all requests covered, and the courier can returns to the warehouse.

Infeasible Tour
A tour that cannot satisfy one or more rules, for example: a courier cannot reach all addresses or the pickup/delivery order cannot be respected.

Rejected Request
A delivery request that cannot be assigned to any courier. It is marked as rejected by the dispatcher.

Active Tour
A tour currently being executed by a courier.

Completed Tour
A tour successfully executed by a courier, ending at the warehouse.

5. Overview of the Model

The **model** represents the core data structure of the application. It defines all the entities involved in the management of bicycle delivery tours within a city. Each class corresponds to a real-world concept: intersections and streets form the map, delivery requests represent customer orders, couriers execute deliveries, and tours group all operations assigned to a courier.

1. City Map

The city map is the central element of the system. It models the **urban road network** as a graph: intersections are the nodes, and road segments are the edges that connect them. Each segment is directed and has a defined length and street name. This structure enables the system to determine routes between any two points in the city based on distance and estimated travel time. The map also contains the list of available couriers, delivery requests, and an adjacency structure used to compute optimal routes.

2. Intersection

An intersection represents a **precise geographical point** in the city. It corresponds to a crossroad, junction, or roundabout. Each intersection is identified by a unique number and defined by its latitude and longitude. Intersections serve as reference points for deliveries and as endpoints of road segments.

3. Road Segment

A road segment connects two intersections in one direction. It represents a **street portion** that couriers can travel through. Each segment contains:

- a starting intersection,
- a destination intersection,
- a length (in meters),
- a street name,
- and an estimated travel time based on the courier's constant speed (15 km/h).

These segments form the foundation for computing travel paths and travel durations between addresses.

4. Courier

The courier represents the physical person performing deliveries by bicycle. Each courier has:

- a unique identifier,
- a name and a phone number,
- and a current location (generally the warehouse at departure).

Couriers are assigned one or several tours according to workload. In the system, the courier embodies the **executor of the generated route**.

5. Delivery Request

A delivery request corresponds to a **customer order**. It specifies:

- a pickup location,
- a delivery location,
- and the service times required at each stop (pickup and delivery).

These requests are received by the dispatcher and imported from XML files. Each one is later converted into a planned **delivery** integrated into a courier's tour.

6. Delivery

A delivery represents the **operational form** of a delivery request integrated into a tour. It connects concrete objects of the model:

- the pickup and delivery intersections,
- the courier assigned to perform the task,
- and the service times and planned departure hour.

Each delivery constitutes one step of a courier's route.

7. Tour

A tour corresponds to the **complete route** assigned to a courier. It gathers:

- all deliveries to be completed,
- the total travel distance and time,
- the total service time,
- and the start and end times of the journey.

Each tour starts and ends at the warehouse, by default at 8:00 a.m. Whenever a new delivery request is added, the system recalculates the tour to minimise the courier's return time to the warehouse.

8. Adjacency Structure

To facilitate route calculation, the map builds an **adjacency list**. Each intersection is associated with the list of neighbouring intersections that can be reached directly by a road segment. This structure allows for efficient exploration of the road network and the computation of optimal paths between multiple locations.