Better Than WAZE

High Level Design

Submitted by:

Tomer Kay

Eylon Shoshan

Ran Yehezkel

Dor Granat

Liran Farhi

Yair Feldman

Supervised by:

Tom Palny

Table of Contents

Abstract

Better Than WAZE (BTW) is a revolutionary city navigation application. It is built for the comfort of the citizens, and managed by the municipalities. BTW provides services that are similar to those of popular navigation applications, such as Waze or Google Maps, but offers a significant advantage - the ability to control the traffic lights in real-time. This functionality can prove vital in an age where self-driving autonomous cars are gaining popularity and slowly changing the face of modern transportation.

1. Introduction

1.1 General Project Description

In this project we are going to implement a city navigation application. The purpose of this application is to provide drivers in the city with the fastest route from one location to another. In our project, we focus on optimizing the driving routes by analyzing the loads on the different traffic lights in the city. By getting real-time information about the state of the traffic lights, we can give a good approximation of the driving time each route will take. Furthermore, we intend to optimize not only the driving routes, but also the traffic lights themselves, such that the green light will be shifted intelligently between the different directions according to the current load on each road. In addition, every municipality will be able to control and manage the city's road map. We believe that smart cities which use this application will be able to provide their citizens with a quality of life like they have never experienced before, especially with the increased popularity of self-driving autonomous cars, with which the application can exchange real-time data and instructions in order to further improve the city's transportation solutions.

In general, our project involves four different implementation parts. The first is an interface for the usage of the municipality. The second is an interface for the usage of the drivers - a navigation mobile app. The third is the "brain" behind the applications, which includes all of the different algorithm we will use for solving the various optimization problems. The fourth and final part is the visualization part, where the work of the first three parts can be fully expressed.

1.2 Programming Environment

We will use three different platforms for implementing our project. The main programming language we will use will be Java. The coding environment will be IntelliJ IDEA, which is a Java integrated development environment for developing computer software,

developed by JetBrains. Also, we will use Github for sharing the work between the group members and keeping up with each other's progress.

2. Theoretical Background

2.1 Shortest Paths in Graphs

The problem of finding the best route for a driver from point A to point B could be naturally modeled to shortest paths problem on graphs- the vertices are points on the road and directed edges if there is road between them.

There are several algorithms to solve the shortest paths problem. The main ones are: Bellman-Ford, Dijkstra, Floyd-Warshall, Johnson.

We present Dijkstra since it plays a role (but not the main one) in our solution. Dijkstra finds the shortest paths from a single source vertex. The running time is: O(|E|log|V|) and could be even reduced to O(|E|+|V|log|V|). The weights on the edges must be non-negative. The algorithm does a relaxation on the edges of the graph but this time each edge is relaxed only once. The next edge (u,v) to be relaxed in each iteration is chosen by the minimum value of distance(u).

However the specific problem of our project is a bit different. We have to support real-time routing, meaning we have to dynamically optimize drivers' routes during their driving.

Our approach is to use one of two options OSRM (c++) or GraphHopper (java) as an infrastructure and extend it accordingly. Those platforms implement **Contraction Hierarchies** (CH) which is a technique to speed up shortest-path routing by first creating precomputed "contracted" versions of the connection graph. It can be regarded as a special case of "highway-node routing". Contraction hierarchies can be used to generate shortest-path routes much more efficiently than Dijkstra's algorithm or previous highway-node routing approaches, and is used in many advanced routing techniques. CH also applies Dijkstra, throughout the algorithm.

Moreover, OSRM and GraphHopper allow to **integrate real-time data** into the process of routing. Thus, we can take advantage of it in order to add support for traffic lights based routing.

2.2 Shortest Paths with Partial Output

Dijkstra and Floyd-Warshall algorithms has an interesting property: They can find the shortest paths with maximal length k (for some k). In Dijkstra's case we can stop if we reach a certain length / weight, and in Floyd-Warshall's case we could choose to calculate d_k for k < |V|.

Note that the algorithms do not necessarily have to stop, but can return a partial output during their runtime.

This idea could help us if we want to reduce the runtime of the algorithm and output a partial path to the destination. When the driver reach the end of the partial path he will request a new path. Thus the driver doesn't have to wait to the shortest path algorithm to fully stop.

3. Basic System Functionalities

The central essence of our project is to supply an interactive navigation application, which basically suggests the most optimized way to get from one place to another in a given city. The best route will be calculated based on real-time parameters of the traffic, and mostly based on traffic lights scheduling.

In addition, we offer to each city which our system is going to be installed in, a vast functional management station. The station would allow them to get live traffic statistics, manage the roads in high resolution of functionality, and customize the city map freely.

We also include amazing built-in features that would improve the driving experience in the city, such as dynamic self-learning traffic lights scheduling, friendly graphical user interface both for drivers and the central management station.

4. Software Implementation

4.1 Top-Level View & Modules

Four central components are the actual core of our project, as described below.

First of all, we present the **visualization engine**, as it plays a role in the other project components. This engine provides an interface in order to get a real-time information about the traffic in a city. This engine processes this data, and present it on a real map of that city. It is also visualize the status of traffic lights all over the city, movement of vehicles, loads on the roads, and other live data that is relevant for the municipality control station. In addition, the same engine also provides a baseline for the navigation mobile app, as it produces the visualization on a map, as well as the optimized way which is suggested to the driver.

- Input: Map representation, traffic and traffic lights states for a given time. The data should be streamed on real-time.
- Output: Visualized representation of all this data on a map. That view is being updated on real-time.

Now, without any further delay, we will move forward to the central three components:

- **User Navigation Mobile App** - Each driver who wants to use our services is requested to download a dedicated navigation application to his mobile smartphone. The app offers a friendly interface for navigation, searching, etc.
    - Dynamic Routing:
        - Input: A driver enters its destination. The application also updates the server about the driver location, with real-time data.
        - Output: The app will communicate with the server, and provide the best route to the user. The routing is also being optimized dynamically, based not only on traffic loads but also on traffic lights scheduling.
- **Municipality Control Station** - In every city that our system is installed, there is a central station that controls the general activity.
    - Controlling City Map:
        - Input: The controllers enters commands, such as: add/remove/temporarily-disable traffic-light/road.
        - Output: City map is updated accordingly.

- Getting Live Statistics:
  - Input: Sub-area of the city is entered by the supervisor.
  - Output: Traffic lights statuses and general statistics from roads in the requested area is informed and presented on screen.

- **Backend Routing Machine** - The routing algorithm, which is detailed in the theoretical section above, mainly find the best routing for drivers and also balancing the loads on roads by optimizing traffic lights scheduling.
  - Input: Live data of current location, a requested destination, traffic loads and traffic lights scheduling.
  - Output: Optimized routing for the driver for that specific interval of time. In addition, the system learns the loads and balances the traffic lights scheduling accordingly.

Additionally to these three main components, we present another vital component, which is going to be used mainly for the matter of testing and demoing. The **Traffic Simulator**. generates a dummy traffic on a city. The simulation can be adjusted using several parameters, such as traffic level, choosing origin and destination of driving session, and so on.

- Input: Requested area on a map, traffic level, origins and destinations of driving cars.
- Output: Simulated traffic on the requested area.

4.2 Main Menu

4.2.1 User Interface

4.2.1.1 City Side

In the city side we can manage & monitor the traffic in the specific location.

As we can see in the following screen, first we select a location (city & country).

Our actions are:

- Statistics of specific road/traffic light/junction.
- Real time information
- Adding an element to the map

- Disable/enable road/traffic light/junction.
- Export report

The map will composed the following elements:

- **Traffic light:** marked by large point. The color of the point represents the status of the traffic light.
- **Junction:** marked by small black point.
- **Cars:** marked by cars icon. The number next to the icon is the number of neighbors cars. The color represent the load - from green to orange and red.
- **Road:** marked on the map.
- **Disable:** we can disable junction/road/traffic light. marked by disable icon.

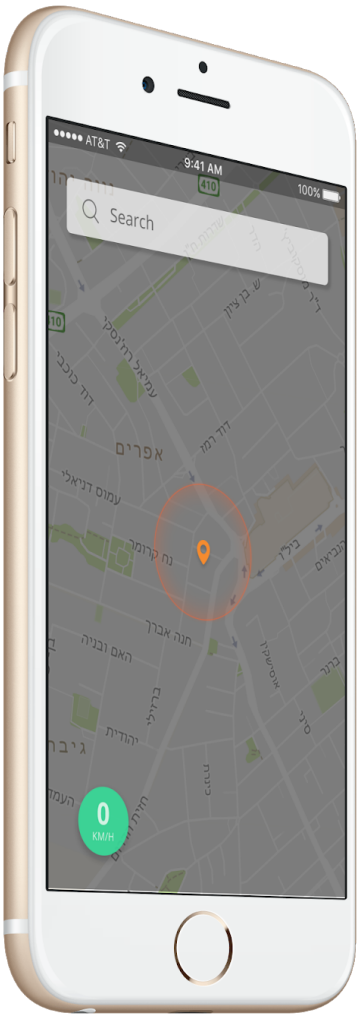
We can add traffic light/road/junction as in the following screen:


Israel
Tel Aviv
Average traffic load: 5
Traffic light: 5
Export Report
Statistics
Real time
Add element
Disable/enable element
Add element
Traffic light
Road
Junction

4.2.1.2 User Side

In the user side we will be able to navigate specific location.
To navigate to specific location the user:

- Search for specific location. We can filter by type of place, such as: restaurant, parking, gas station, supermarket and more.
- Select location from list of places sorted by distance from our current location.



4.2.2 Features

4.2.2.1 Customer Side

- Our main feature for our customers is receiving the best path for their destination in the best way they would want. Best path could be defined by the customer as:
    - Minimum time
    - Favorable conditions as in driving with the least amount of traffics
    - Driving in any kind of roads or rather only on proper roads
    - Shortest path
    - Driving mainly through main roads (this way the customer could learn the path by heart)
    - Driving near special attractions with explanation on them
- Searching for the nearest common destinations as:
    - Restaurant
    - Petrol station
    - Mall
- Save preferences as home, work's place and more tags to be named by the customer desire.
- Save the history of past destinations

4.2.2.2 City Municipality Interface

- City Municipality can update our software of any change in the field although our system is being updated by itself. City Municipality can notify on:
    - New road
    - Roads not available at the moment
    - Gates to villages that are closed on certain hours
    - New light at a junction
    - New Junction by his type - circle/lights/stop signs
- City Municipality can ask for Real Time information about the roads such as jammed and free to ride rodas.
- City Municipality can ask for statistics on variety of properties belonging to the roads:
    - Popularity of a road
    - Traffic times of a road
    - Speed average of a road in a particular time of the day
    - How many cars drive through a junction in a particular time of the day
    - Most jammed junctions and roads

5. References

- OSM - https://wiki.openstreetmap.org/wiki/Main_Page