Incident Management System
For Intelligent City Development
28 September 2018

Group 2
Supervisor: Dr. Cheng, Reynold C.K.
Chow, Terence Yin Chak
Kwong, Matthew Wang Shun
Yuen, Calvin Hoi Man
Background

Today, traffic congestion has become one of the biggest problems in many cities including Hong Kong. In fact, according to the Transport and Housing Bureau, Hong Kong citizens can suffer up to 200% increase in commuting time because of traffic congestion [1]. A lot of valuable time is wasted and may lead to huge economic losses. As incidents like car crashes and nearby large-scale events are the main contributors to severe traffic congestion, a service that provides route suggestions to general public like drivers that minimizes the route path segment in the congested areas may shed light on the solution towards this problem.

On top of routing away from traffic incidents, this project also put focus on the crowd flow control for large-scale events and occasions like Hong Kong Book Fair and Friday night Lan Kwai Fong. In these settings, large number of road users are trying to find quickest routes to leave the same location to their destination at the same time. If these route-searching requests are not handled properly and are returned with the same quickest path, congestion on the relevant path segment may be created and the paths suggested will no longer be the quickest. Therefore, crowd-based routing will be investigated and implemented to serve mass requests such that the routes suggested can still be efficient enough for users to reach their destination.

Currently, there are several navigation and routing applications and SDKs floating on the market which aim to solve this problem. One of the examples is TallyGo [2], previously known as ClearPath, which provides a Google Map like navigation system while integrates real-time accident data into it. Users can make origin-destination requests (O.D. requests) to the application service and a suggested route with shortest travelling time is returned to the users and is presented on an application front-end. For TallyGo, incident and congestion data are incorporated with route suggestion, in which alternative routes will be suggested so that the congestion can be remediated by re-routing traffic flow to other paths.

However, there are several limitations that prevent these applications or SDKs from directly applied to Hong Kong in general:

1. These applications and SDKs only support routing and incident management in limited cities and countries. In most cases, Hong Kong is not included.

2. Some of these applications and SDKs require extra infrastructures. In the case of TallyGo, sensors have to be installed on the road to provide accurate, up-to-date data but this may not be feasible for the scope of this project.

3. Requests from the same origin may be routed to paths with the same initial path segments. When large-volume requests are made, extra congestion may be introduced at overlapped path segments, which diminishes the benefits of such applications.
Basics

Traffic incidents refer to any kind of incidents and situations which have potential negative impact to the traffic, for example car accidents and protests. Major events like music concerts and Hong Kong Sevens are also considered as Traffic incidents although the events themselves are not, as the huge crowd of audience trying to leave the venue after the events may cause traffic congestion.

Crowd-based routing refers to an extension of shortest path routing, which avoids creating extra traffic congestions caused by directing large amount of queries to the same road segment. Consider the scenario that a large number of vehicles are attempting to cross the Cross-Harbour Tunnel from Hung Hom to Causeway Bay during rush hour. Each vehicle may have distinct origin and destination but all of them arrive at Cross-Harbour Tunnel simultaneously. This causes traffic jam at the tunnel. Crowd-based routing aims to prevent such scenarios.

Objective & Scope

The objective of this project is to provide a service for Hong Kong traffic navigation and routing while taking incidents and mass O.D. requests into consideration. As a result, users can shorten their travelling time by avoiding existing traffic jams without creating new congestions. The service aims to achieve the following:

1. Provide a road network shortest path algorithm, which:
   a. Supports high performance query on a complex map.
   b. Supports certain degree of real time update to the graph structure depends on the road network condition
   c. Achieves crowd-based routing. Users will be distributed to different paths and road segments.
   d. Takes the incidents into consideration. Incident-affected or congested road segments will less likely to be included in the suggested path.

2. Receive incidents detected and reported by other services. Incident simulation will also be investigated and implemented to inject incidents into the routing algorithm.

3. Provide an application which:
   a. Notify users when new incidents appear
   b. Provide an graphic user interface (GUI) which enable users to select the desire origin and destination, and display the path suggested.
   c. Track and display users’ GPS location
Methodology

The Iterative Cycle

Figure 1: The Iterative Cycle [3]

This project employs an Iterative Cycle (see Figure 1) approach for the software product development. The cycle will keep repeating until the final product is delivered. The main advantage of such practice is that it encourages modularization of the software. Each iteration results in manageable working executable (module) which allows easy testing and review for the workability and correctness in an earlier stage. Risks like failing to solve difficult problems such as crowd-based routing can also be identified and mitigated earlier by doing unit testing frequently for each component as well, because smaller modules can be tested earlier than a large-scale full-fledged project.
System Architectural Structure

The project tries to deliver a service using server-client architecture with an API hosted on the Amazon Web Service (AWS) EC2 platform for the client to communicate with the server (see Figure 2). As described on Figure 2, the backend incident management system depends on modules for route computation, graph update and incident analysis instead of having these feature hard-wired in the backend server code base. This can ensure segregation of duties and modularization among different components so that testing and maintenance can be easier and the development work can be distributed more efficiently among the team. More importantly, it allows pluggable modules as microservices for easy module replacement like switching routing algorithms. Details in each component are unfolded as follows:

OpenStreetMap
This project utilizes map data provided by OpenStreetMap (OSM) to model the graph structure for path finding. OpenStreetMap (OSM) is a collaborative map project which provides free and editable geographical information. Its map data are crowdsourced and available to the public under the Open Database License (ODbL). With its editability and ODbL, OSM allows this project to re-create new routing service by altering the path weights, which is an important functionality needed for crowd-based routing.
Routing Module
Routing Module (RM) is responsible for finding the quickest path corresponding to the requested origin and destination, as well as building and maintaining the graph structure needed for the shortest path searching from the map data provided by OSM. In this module, the road network algorithm plays a significant role, which has the direct impact to the performance of the overall system. One RM is used to represent one type of routing algorithm.

Graph Structure
Instead of dealing with graph data structure for maps in a separate module, RMs are also responsible for building the graph structure for path finding as well as updating the graph according to incidents and crowd requests. Majority of the algorithms include graph construction as a substantial part of their optimization processes and the way they build and work on a road network is different from each other. As the result, the graph structure for optimal performance of each algorithm varies. Therefore, each RM will have its own Graph Building and Maintenance implementations such that the graphs will always be optimal for the corresponding algorithm.

Crowd Control Module
Crowd Control Module (CCM) is responsible achieving evenly distribution of the load of the road network by updating the map graph. In some situations, shortest path queries cannot be considered independently ignoring the fact that multiple queries can be requested at the same time. When multiple queries are routed to the same shortest path, this path might be congested and no longer be offers the shortest time for the user to reach their destination. Therefore, an algorithm is needed to distribute the load of the network.

Incident Detection System
Incident Detection System (IDS) is responsible for detecting incidents through machine learning and keep the Incident Analysis Module updated. When new incidents are detected, it fetches an incident report and send it to the Incident Analysis Module. The incident report include information necessary to identify and locate the incident, for example the date and time when the incident happens, and the latitude and longitude of the incident.

This system is followed by peer team, which is not within our scope for this project. In this case, a hook API will be exposed for the IDM to post incident information to our service.

Incident Analysis Module
Incident Analysis Module (IAM) is responsible for analysing the impact of the incidents and calculate changes to be made to the graph structure maintained in RM if necessary. The analysing algorithm in this module plays a major role to the overall system by initiating graph updates to capture incidents and abstract them in the graph in a timely manner to route users correctly.
Incident Management System

Incident Management System (IMS) is the core of this system which depends on the implementations of RM, CCM and IAM. This module coordinates the modules as a controller and as the result provides functionality to the service.

Server-Side API Endpoints

Taking reference from similar products and services like Google Maps, it is likely that this system will eventually become a web service when deployed as a commercial product so that public users can access it through the Internet. In the scope of this project, a client-side mobile application will be developed and therefore a set of server-side API endpoints is needed as a middleware for the application to make requests to the IMS for the O.D. requests and incident alerts.

Client-Side Application

A mobile application will be delivered for users to send O.D. requests and view the returned route, as well as for the back-end service to track the users’ GPS location for route navigation and incident notification.

Traditionally, mobile application development are completely separated by the Android and iOS platforms. However, as cross-platform mobile development languages and technologies evolve, this project chooses to develop a cross-platform application. The development process can be hugely simplified as the application components can be reused and the need to re-develop the same functionalities for different platforms is eliminated.

Crowd and Incident Simulation

For the sake of project demonstration, the client-side application will also be responsible for performing crowd and incident simulation.

Crowds will be simulated as huge amount of O.D. requests with same origin or origins very close to each other but different destinations being fired at the same time. Real-life statistics like attendance of large events (e.g. attendance of a concert at Hung Hom Auditorium) can be taken into reference for the number of O.D. requests to be generated and the location of origin.

As a baseline for the simulation, incidents will be injected into the map graph at random time and random location via the API webhook for incident reporting. If real-life incident report can be obtained from the peer group, the simulation of incidents can be done with real-life data.

Multiple O.D. requests with same origin will be sent to the backend server to simulate a crowd. Routes taking the ongoing incidents into account for each request will be returned to the application. The application will then show the distinct routes and simulate the crowd will move according to the routes suggested.
Mapbox

Mapbox is a provider of custom online maps which offers SDKs and APIs and allows developers to use them for both mobile and web development of new projects and services. It is going to be used for the display of maps and visualization of the user’s current location and the suggested path.

The main reason of choosing Mapbox instead of Google Maps is that the terms of services of Google Maps forbids the creation of “a substitute of the Google Maps Core Services” [4]. Unfortunately, since shortest path routing is one of the main features of Google Maps, it cannot be used for this project. Despite being relatively less popular, Mapbox is highly customizable. One can even download and customize the map data for individual needs and offline usage.

Amazon Web Services EC2 (AWS)

AWS EC2 is a Platform-As-A-Service (PaaS) product that provides on-demand computational resources and platform for development and product deployment. Such service is chosen for this project because it is expected that the allocated development server may not offer enough computational power and main memory for the Routing module.
Algorithms

In this project, a total of three sets of algorithms will be needed for the RM, CCM and IAM modules. For each module, different existing algorithms will be evaluated and one will be selected for the implementation of the final delivered service. In cases where none exists, a new algorithm will be proposed for that module instead. Modification of the selected algorithms may be done in order to better facilitate the functionality of the system.

Algorithm for the Routing Module

The algorithm set used by RM will consist of graph building, graph maintenance and path finding. At graph building and graph maintenance stage, the original map data will be processed into suitable abstract data structure such that a real time query can be achieved. At the path finding stage, the path-finding algorithm answers O.D. requests from user. Some algorithms require additional post-processing to translate results from initial calculation results to the actual path for the user. Such approach has the advantage of a better search speed than normal methods, with the cost of a longer pre-computation.

The basis of the algorithm design for the final product will be selected among variations of Dijkstra-based algorithms, examples being Multi-Layer Dijkstra (MLD) [5] and Contraction Hierarchies (CH) [6]. The selected algorithm will be the determining factor of how the road network is represented within the system, and what additional data to be stored in order to speed up the calculation process, e.g. by caching. The selection will be based on the performance, ease of implementation, and how the algorithm support the algorithm design of other modules.

Algorithm for the Crowd Control Module

Depending on the basis algorithm chosen, the algorithm used by the CCM will perform certain kinds of modifications to the map graph or the O.D. request when multiple O.D. requests interfere with each other. For example, CCM may provide graph modification parameters to the graph maintenance component in RM. Modification of O.D. requests may also be made in the process if needed. The basis of the algorithm design will be developed with some concepts borrowed from different evacuation routing algorithms.

Algorithm for the Incident Analysis Module

The algorithm used by the IAM will provide graph modification parameters the the graph maintenance component in RM when incident is reported by IDM.
Schedule and Milestones

**Week 1 - 4 (September 3 - 30, 2018)**

*Algorithm Researching and Planning*
- Research on spatial road network graph modeling
- Research on existing data
- Research on algorithm
  - Routing
  - Crowd control

*Software Design I and Setup*
- Building general front-end and back-end template
- Research on Mapbox SDK
- Research on Amazon Web Services

*Phase 1 Deliverables (September 30, 2018)*
- Project Plan (this document)
- Project Web Page

**Week 5 - 13 (October 1 - November 30, 2018)**

*Data Preparation, Algorithm Design and Implementation I*
- Create a simple graph as preliminary example; Use a small-scale road network map (e.g. one of the Hong Kong Districts) for testing.
- Research and Implement incident-aware routing algorithms set.
- Unit tests for correctness and effectiveness of algorithm continuously.

*Software Implementation I*
- Implement prototype of backend API
  - Receive and handle incident reports; Receive O.D. requests and response.
  - Create wrappers for the modules to be used on the back-end server as add-ons.
- Implement frontend application
  - Display the map.
  - Makes O.D. requests to the backend API.
  - Draw the path on the map received from backend API.
- Unit tests for functionality of the implementation continuously.
Incident Management System - Project Plan

Week 14 - 17 (December 1 - 30, 2018)

Algorithm Design and Implementation II
- Research and implement a baseline Crowd Control algorithm.

Software Implementation II
- Integrate the RM and IAM add-ons with the backend-service.
- Add UI components to frontend
  - Searchbox
  - Settings Menu
- Unit tests for functionality of implementation continuously.

Integration Testing I
- End-to-end testing for inter-module and server-client communication.

Phase 2 Deliverables I and First Presentation Preparation (December 17 - 31)
- Prepare interim report document
- Prepare first presentation materials, e.g. slideshow, frontend and backend application demo

Week 18 (December 31, 2018 - January 4, 2019)
- Practice presentation
- Buffer for overrun in plan

Week 19 (January 7 - 11, 2019)
- First presentation

Week 20 (January 14 - 18, 2019)

Algorithm Design and Implementation II
- Continue the research and implementation of the Crowd Control algorithm.

Software Design II
- Research on Text to Speech in mobile application

Integration Testing II
- Exploratory testing

Phase 2 Deliverables II (January 20, 2018)
- Preliminary implementation
- Detailed interim report
Incident Management System - Project Plan

Week 21 - 23 (January 21 - February 8, 2019)

*Algorithm Design and Implementation II*

- Continue the implementation and improvement of the baseline Crowd Control algorithm.
- Unit tests for the correctness and efficiency of the algorithm continuously.

*Software Implementation III*

- Acoustic turn by turn navigation

Week 24 - 26 (February 11 - March 1, 2019)

*Software Implementation IV*

- Incident feedback and analysis module
- Unit tests for functionality of implementation continuously.

Week 27 - 32 (March 4 - April 12, 2019)

- Work on final report.
- Prepare the banner for final exhibition.
- Prepare final presentation materials, e.g. slideshow, frontend and backend application demo.
- Practice presentation.
- Keep testing and debugging the overall application.
- Buffer for overruns in previous plans.

Week 33 (April 14 - April 19, 2019)

*Phase 3 Deliverables (April 14)*

- Finalized tested implementation
- Final report

*Final Presentation (April 15 - 19)*

Beginning of Week 35 (29 April, 2019)

- Project exhibition
## Division of Labour

As a relatively-large amount of investigation and development work are involved in this project, the works will be distributed among the team members as follows:

<table>
<thead>
<tr>
<th>Team Member</th>
<th>Responsible Work</th>
</tr>
</thead>
</table>
| Chow, Terence | ● Study on working mechanisms of existing routing algorithms.  
  ○ Analyze the complexity, pros and cons of each routing algorithm.  
  ● Assist Calvin on the implementation of the algorithms.  
  ● Design the system architecture of the service.  
    ○ Study and select from different technologies that can be used for this project. Determine how they interact.  
  ● Implement of backend server in Node.Js.  
    ○ Create Node.Js wrappers for the C++ implementations of the RM, CCM and IAM modules to be used as addons. |
| Kwong, Matthew| ● Implement of backend server in Node.Js.  
  ○ Create and expose API endpoints linked to IMS.  
  ● Implement of front-end mobile client application in React Native.  
  ● Study on the usability and reliability of the current data.  
    ○ For simulation of crowds and incidents, as well as the testing of created algorithms. |
| Yuen, Calvin  | ● Study on working mechanisms of existing routing algorithms.  
  ○ Analyze the complexity, pros and cons of each routing algorithm.  
  ● Study on existing evacuation algorithms.  
    ○ Analyze and select reusable elements from these algorithms to be put into our solution.  
  ● Implement the algorithms for the final service in C++. |
References


