ROUTE PLANNING ALGORITHMS FOR URBAN ENVIRONMENT

OW YI XIAN

(B.Eng.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2012

DECLARATION

ACKNOWLEDGEMENTS

I thank my caring parents and girlfriend who have always been there to provide moral support. I give my sincere thanks to both of my supervisors Dr. Mandar Chitre and Dr. Daniela Rus for accepting me as their student and for giving me this much appreciated research opportunity especially to Dr. Mandar for his close guidance throughout my candidature. I give my thanks to the kind people of Future Urban Mobility, Singapore-MIT Alliance for Research and Technology and Acoustic Research Laboratory, National University of Singapore for all of their assistance and support. These people include Dr. Zou Bingran, Mr. Lim Sejoon, Mr. Mohanan Panayamadam, Mr. Tan Yew Teck, Mr. Koh Wee Kiat, Mr. Shankar Satish to name a few. Thanks to all for moral support and best wishes.

Contents

Abstract

This thesis introduces solutions to two route planning problem for urban environment. The first is the *probe allocation problem* where a team of agents is deployed to monitor traffic condition in an urban environment such that accurate estimation of traffic condition along all roads can be made. The second is the *bus route planning problem* where public transportation is optimized such that commuter travel time is minimized. The performances of these solutions are studied both analytically and through simulations.

Surveillance problem in a variety of scenarios such as hostile environments, environmental monitoring and law enforcement missions (i.e. border patrol) etc have been widely studied. However, monitoring of traffic conditions in urban environments by deploying a team of agents is a novel problem not found in any existing literature. This problem is termed as the probe allocation problem. In light of the recent developments of automated cars, studies of such surveillance problems in urban environments will prove to be useful. Two sets of algorithms are proposed for this problem, each pertaining to a different deployment strategy.

In addition, it is shown that the probe allocation problem is very similar to the bus route planning problem. In the bus route planning problem, bus routes are being optimized such that the total travel time of every commuter is minimized in addition to other factors such as the number of transfers made etc. A novel algorithm (a modified version from one of the algorithms originally proposed for the probe allocation problem) is proposed for the bus route planning problem. The proposed algorithm for the bus route planning problem outperforms many existing bus route planning solutions when compared against the well known Mandl's Benchmark.

List of Tables

List of Figures

List of Symbols

G	Directed graph denoteing road network
V, V_j	Set of vertices of G, jth vertex in $V : 1 \leq j \leq \|V\|$
E, E_i	Set of edges of G, ith edge in $E : 1 \leq i \leq \|E\|$
t	Time of the day
τ_{E_i}	Average time taken to travel along edge E_i and transit to another edge
$\mu_{t,E_i}, \sigma^2_{t,E_i}$	Expectation and variance from past information on edge E_i at time t
$\bar{\mu}_{t,E_i}, \bar{\sigma}^2_{t,E_i}$	Estimated expectation and variance on edge E_i at time t
$\hat{s}_{E_i}, \check{s}_{E_i}$	Max. and Min. of average traffic speed on edge E_i.
λ_{t,E_i}	Total probability of $\mathcal{N}(\mu_{t,E_i}, \sigma^2_{t,E_i})$ from $\check{s}_{E_i}$ to $\hat{s}_{E_i}$
$\bar{\lambda}_{t,E_i}$	Total probability of $\mathcal{N}(\bar{\mu}_{t,E_i}, \bar{\sigma}^2_{t,E_i})$ from $\check{s}_{E_i}$ to $\hat{s}_{E_i}$
$p^s(s; t, E_i)$	Probability function for average speed on edge E_i at time t
$g(E_i, t), \hat{g}(E_i, t)$	Actual value and estimated value of average speed on edge E_i at time t
$\tilde{g}(E_i, t), \tilde{g}^{\max}(E_i, t)$	Entropy and maximum entropy of average speed on edge E_i at time t
β_{t,E_i}	Rate of change of $\tilde{g}(E_i, t)$
f_{E_i}	Frequency at which edge E_i must be visited by an agent
$f^{\max}, f^{\min}$	Maximum and non-zero minimum frequency out of all f_{E_i} values
$k_{\max}, k_{\text{opt}}$	Total number of groups $\gamma_k : 1 \leq k \leq k_{\max}$, Optimum total number of groups $\gamma_k : 1 \leq k \leq k_{\text{opt}}$
f'	Frequency interval defined as $f' = \frac{f^{\max} - f^{\min}}{k_{\max}}$

γ_k	Set of edges grouped into k^{th} group with respect to f_{E_i} and f'
α_k	A disjoint circuit for k^{th} group such that it is a subset of edges in E and forms a subgraph of G consisting of one or more components, such that there exists an Eulerian circuit in each of these components.
α'_k	Subset of γ_k such that $\alpha'_k \cap \alpha_{k'} = \emptyset \mid \forall k' > k$
$f_k^{\max}, f_k^{\min}$	Maximum and minimum f_{E_i} of all edges $E_i \in \gamma_k$
$D(\alpha_k, V_j)_{\text{in}}, D(\alpha_k, V_j)_{\text{out}}$	Number of in coming and out going edges incident on vertex V_j in subgraph formed with edges in α_k
$D^{+/-}(V_j)$	The result from the difference between the in-degree and out-degree of node V_j
V_j^+	Set of nodes V_j such $D^{+/-}(V_j) > 0$
V_j^-	Set of nodes V_j such $D^{+/-}(V_j) < 0$
α''_m	A circuit such that it is a subset of edges in E and forms a subgraph of G consisting of exactly one component which has an Eulerian circuit
k_m	Index indicating $\alpha_{k_m} : \alpha''_m \subseteq \alpha_{k_m}$
f_m	$f_m = \max(f_{E_i}) \vert E_i \in \alpha''_m \cap \alpha'_{k_m}$
τ'_m	Total time needed to traverse Eulerian circuit of subgraph of G formed by edges in α''_m
$n_m, n_{m,m'}$	Number of agents required by α''_m and combination of α''_m and $\alpha''_{m'}$
$V(\alpha''_m)$	Set of vertices incident to at least one edge in α''_m
n_{E_i}	Minimum number of agents needed on edge E_i evenly spread out
n'_{E_i}	Minimum number of agents needed on edge E_i so that there are n_{E_i} number of agents evenly spread out on edge E_i
$n^{\text{h}}_{E_j}$	Modified version of n'_{E_i} such that no value is equal to zero
n''_{E_i}	Actul number of agents on edge E_i
a	Confidence level that the goal/objective is met
c_{E_i}	Road capacity of edge E_i
P, P'	Both are equilibrium probability of a Markov Chain

p_{E_i}, p'_{E_i}	Equilibrium probability that an agent is on each edge E_i
T, T'	Both are transition matrix of a Markov Chain
$t_{E_i,E_j}, t'_{E_i,E_j}$	Probability for an agent on edge E_i to choose to move to edge E_j
$S^{\mathrm{A}}(E_i)$	Set of out-going edges incident to the entrance node for edge E_i inclusive of edge E_i
$S^{\mathrm{B}}(E_i)$	Set of out-going edges incident to the exit node for edge E_i
$p^{\mathrm{B}}_{i,j}, P^{\mathrm{B}}$	Representing number of commuters travelling from bus stop i to j, demand matrix
$v^{\mathrm{B}}_i, V^{\mathrm{B}}, V^{\mathrm{B}\prime}$	Vertex representing bus stop i, set of all vertices, set of vertices corresponding to bus terminals
$e^{\mathrm{r}}_{i,j}, E^{\mathrm{r}}$	Edge representing road between bus stop i and j, set of all edges $e^{\mathrm{r}}_{i,j}$
$e^{\mathrm{w}}_{i,j}, E^{\mathrm{w}}$	Edge representing walking path between bus stop i and j, set of all edges $e^{\mathrm{w}}_{i,j}$
$c^{\mathrm{r}}_{i,j}, c^{\mathrm{w}}_{i,j}$	Time taken to traverse the road/walking path respectively
$e^{\mathrm{B}}_{i,j}, E^{\mathrm{B}}$	Results due to combination of $e^{\mathrm{r}}_{i,j}, e^{\mathrm{w}}_{i,j}$, set of all $e^{\mathrm{B}}_{i,j}$
$G^{\mathrm{r}}, G^{\mathrm{w}}, G^{\mathrm{B}}$	Graph consisting only of edges in $E^{\mathrm{r}}, E^{\mathrm{w}}, E^{\mathrm{B}}$ respectively
$\bar{p}$	Mean of all non-zero $p^{\mathrm{B}}_{i,j}$ values
$c(G^{\mathrm{B}})$	Function that returns the number of components in graph G^{B}
$c^{\min}(G^{\mathrm{B}})$	Function that returns the number of components in graph G^{B} excluding vertices that are not incident to any edge
$e(G, q)$	The set of edges in the q^{th} component in graph G
o_k, d_k	Origin and destination of k^{th} selected OD pair
$\alpha^{\mathrm{B}\prime}_k$	Set of edges in the shortest circuit consisting shortest path from o_k to d_k
$\alpha^{\mathrm{B}}_m, \alpha^{\mathrm{B}}$	A candidate solution before algorithm part 2; an actual bus route solution after. Set of all α^{B}_m
$p^{\mathrm{B}\prime}_{m,i,j}, P^{\mathrm{B}\prime}_m$	Number of commuter expected to traverse edge $e^{\mathrm{r}}_{i,j} \in \alpha^{\mathrm{B}}_m$
$d(\alpha^{\mathrm{B}}_m)$	Maximum degree of all vertices $v^{\mathrm{B}}_i \in V^{\mathrm{B}}$

$d^{V^{\mathrm{B}\prime}}(\alpha^{\mathrm{B}}_m)$	Maximum degree of all vertices $v_i \in V^{\mathrm{B}\prime}$
$\sigma(P^{\mathrm{B}})$	Variance of all non-zero values in matrix P^{B}
$\max(P^{\mathrm{B}})$	Maximum of all values in matrix P^{B}
$\gamma^v(G^{\mathrm{B}}, i, j), \gamma^e(G^{\mathrm{B}}, i, j)$	Set of vertices and edges arranged in sequence with respect to the shortest path from v^{B}_i to v^{B}_j in graph G^{B} respectively
$\gamma^{\mathrm{ew}}_q(G^{\mathrm{B}}, i, j)$	Set of edges of a component of subgraph induced by edges in $\gamma^e(G^{\mathrm{B}}, i, j) \setminus E^{\mathrm{w}}$. Each q refers to each component
$\gamma^{\mathrm{vw}}_q(G^{\mathrm{B}}, i, j)$	Start vertice of subgraph induced by edges in $\gamma^{\mathrm{ew}}_q(G^{\mathrm{B}}, i, j)$
$d(v^{\mathrm{B}}_i, G^{\mathrm{B}})$	Function that returns the out-degree of v^{B}_i in graph G^{B}
$N^{\mathrm{B}}, n^{\mathrm{B}}_m$	Total number of buses available, number of buses assigned to each bus route α^{B}_m
$t^{\mathrm{B}}_{i,j}$	Set of indices of bus rotes in α^{B} that describes the bus routes a commuter have to board to travel from bus stop i to j. In short, sequence of bus to board.
$P^{\mathrm{B}\prime\prime}_m, d_m$	Demand matrix for each bus route α^{B}_m, demand on each bus route
$\tau^{\mathrm{B}\prime}_{i,j}$	In-vehicle travel time for commuter travelling from i to j
$\tau^{\mathrm{B}\prime\prime}_{i,j}$	Walking time for commuter travelling from i to j
$w_m, w_{i,j}$	Bus wait time for bus route α^{B}_m, bus wait time for commuters travelling from i to j
$\tau^{\mathrm{BN}\prime}_{i,j}$	In-vehicle travel time for commuter travelling from i to j based for NUS bus routes
$\tau^{\mathrm{BN}\prime\prime}_{i,j}$	Walking time for commuter travelling from i to j for NUS bus routes
$w^N_{i,j}$	Bus wait time for bus route α^{B}_m, bus wait time for commuters travelling from i to j for NUS bus routes
$\tau^{\mathrm{BP}\prime}_{i,j}$	In-vehicle travel time for commuter travelling from i to j for computed bus routes
$\tau^{\mathrm{BP}\prime\prime}_{i,j}$	Walking time for commuter travelling from i to j for computed bus routes

$w^P_{i,j}$	Bus wait time for bus route α^{B}_m, bus wait time for commuters travelling from i to j for computed bus routes

Chapter 1

Introduction

Much engineering and research effort has been put into in evaluating traffic congestion patterns. The evaluation of traffic condition is important as it allows mitigation of traffic congestions in the short term and it is critical for urban planning in long term. The ultimate benefit of such effort is to increase transportation efficiency [10,12]. The effectiveness of urban planning is essential as we can see from [2] that the amount of resources spent on transportation is high. For instance, it is stated that "About one-third of all city infrastructure investment need is for the transport sector." Cities include Singapore, Hong Kong and Kuala Lumpur etc.

In the *probe allocation problem*, we deploy multiple agents (mobile vehicles) into an urban environment to measure traffic conditions. From [5, 7–9] we know that if sufficient real time and past information are provided, accurate estimations of the traffic conditions can be made. Therefore agents are deployed to provide us with the necessary information. After a measurement is taken, the traffic conditions can vary over time and therefore the uncertainty in traffic conditions increases with time. The objective of the probe allocation problem is to determine the number of agents required and a deployment route for these agents such that the uncertainty in our estimation of traffic speed for all roads will be kept below a certain threshold. The problem can be extended to measure quantities such as human traffic, noise level, air quality and regional temperatures etc which are also relevant to urban planning. However, only the problem of measuring average traffic speed over each road is discussed in this thesis.

The *bus route planning problem* has been widely studied in literature and it involves the optimization of public transport. It is essentially an optimization problem where the objective function consists of a number of factors. Several factors have been known to be considered. These includes passenger waiting time and travel time, number of direct trips, number of transfers made, fleet size, operator cost and profit [20, 21, 23–28]. The objective of the problem is to compute bus routes and bus assignments such that the cost of the objective function is minimized. In this thesis, we identify the similarity between the probe allocation problem and the bus route planning problem. Therefore, we modify one of the algorithms proposed for the probe allocation problem and propose it as a novel approach to solve the bus route planning problem. It is shown that this approach is superior to many existing bus route planning algorithms when compared based on the Mandl's Benchmark.

As the work presented in this thesis is a Singapore-MIT joint project, some of the work in this thesis has been presented as part of Future Urban Mobility's annual workshop. Future Urban Mobility a research group under Singapore-MIT Alliance for Research and Technology (SMART). The workshops included attendees from NUS, MIT, NTU, government agencies such as LTA, etc. In [38], a preliminary study of the probe allocation problem was introduced along with the algorithm presented in chapter 2. In [39], the algorithms in chapter 3 were presented. In addition, work in chapters 3 and 6 have been submitted for the upcoming IEEE Symposium Series on Computational Intelligence (SSCI) 2013 in the IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS) [40, 41].

1.1 Thesis Organization

This thesis is organized as follows. The literature survey is found in section 1.3. In section 1.4, the problem formulation for the probe allocation problem is described detailing the model for traffic speed, entropy of estimation of traffic speed, a formal description of the problem's objective etc. In chapter 2, we show the solution for the probe allocation problem for the special case when all roads are required to be visited by an agent at the same frequency. In chapter 3, the *fixed circuit algorithm* is used to solve the probe allocation problem, giving the intuition behind the solution and their various properties. The problem tackled in chapter 2 assumes that all roads are required to be visited by an agent at the same frequency and is therefore is a special case of the problem in chapter 3 where the assumption is removed.

They are presented in different chapters as their solutions are very different in nature.

In chapter 4, random walk solutions proposed to solve the probe allocation problem are described. In chapter 5, the Singapore scale data and simulation results from Singapore scale simulation for algorithms proposed for the probe allocation problem are shown. In chapter 6, the bus route planning algorithm and its simulation results are shown. The conclusion and proposal for future work are shown in chapter 7. Chapters 2, 3, 4 and 6 includes NUS simulation results found after applying each section's respective algorithms. The simulation setup for the probe allocation problem can be found in section 1.4 and is applicable to the NUS simulations for chapters 2, 3 and 4. The NUS bus route planning problem has a separate simulation setup that can be found in chapter 6.

1.2 Contributions

The main contributions of this thesis will be elaborated in the following paragraphs. They are as follow:

1. Introduction of a novel problem, the probe allocation problem.
   (a) Smaller number of mobile sensors than stationary ones.
   (b) Cost, logistics and maintenance.
2. Different algorithms that solves the introduced problem.
   (a) Fixed circuit algorithm.
   (b) Random walk algorithm.
3. Noted similarities between probe allocation problem and bus route planning problem.
   (a) Modified fixed circuit algorithm applied to bus route planning problem to produce satisfying results.

In this thesis, the novel problem called the probe allocation problem is formally introduced. Two sets of algorithms are introduced to solve this problem with each of them pertaining to a different deployment strategy. The first is called the fixed circuit algorithm. In this deployment strategy, agents are assigned to compute circuits and they traverse it repeatedly. The second is called the random walk algorithm where agents are allowed to move randomly in the road network based on a computed Markov Chain transition probability. A Singapore scale simulation is carried out as a case study to investigate the number of agents required in order to make accurate traffic speed estimations.

The study of the probe allocation problem is especially useful in a number of situations. Its key advantage of having mobile sensors is the fact that a smaller number of moving sensors are required as compared to a large number of stationary sensors. In addition, maintenance of sensors would be more convenient as it is possible to program the agents such that they would return to a specified location either when necessary or periodically; thus lowering cost and logistics of maintenance.

Similarities between the probe allocation problem and the bus route planning problem are identified and have led to the modification of the fixed circuit algorithm so that it is applicable to the bus route planning problem. The algorithm is not only novel but is also deterministic. This is not a common property in bus route planning algorithms due to the complexity of the problem [20–28]. A NUS simulation is carried out and a set of bus routes are computed as a solution for the NUS road network. It is found that the computed bus routes perform better than the current NUS bus routes in terms of total commuter travel time. In addition, simulation is carried out on the Mandl's benchmark and it is found that the proposed algorithm outperforms many existing bus route planning algorithms.

1.3 Literature Survey

In this section, we look at some of the prior work relevant in literature. In section 1.3.1, we discuss the methods used to solve optimization problems. Due to the nature of the problem tackled in this thesis optimization is unavoidable and therefore literature survey on the topic is necessary. The boolean satisfiability optimization was looked into as the selection of edges for routes is most naturally modelled as a binary integer programming problem, for additional details refer to section 1.4 and chapter 3. In section 1.3.2, a brief background on how traffic modelling and prediction is typically done. In section 1.3.3, a specific paper that solves a similar problem to the probe allocation problem is mentioned. In section 1.3.4, some existing algorithms used to solve the bus route planning problem are briefly discussed. No literature survey of mobile agent based monitoring of traffic condition will be discussed as the problem is novel and unique to this thesis. Previous work [35–37], typically assumes stationary agents; for example in [35] agents are modelled as network elements, such as crossings, road segments and routes.

1.3.1 Boolean Satisfiability (SAT) Optimization

In the problems stated in the thesis Abstract and section 1.2, both the probe allocation problem and the bus route planning problem, the need to do optimization is unavoidable. The optimization problems solved in this thesis in particular are all binary integer programming which can be solved as a Boolean satisfiability optimization problem. While application of generic methods is possible, optimization solvers written specifically for binary integer programming problems performs better. For example in [6], we can see the performance comparison between "Pseudo-Boolean Solver version 4" (PBS4) and IBM's ILOG CPLEX which shows that PBS4 is a superior solver as long as the problem is satisfiable. Another example can be found in [17] where SATsolver is used on FPGA routing problems and in [18] where planning problems are solved as satisfiability problems as well. The optimization solver used in this thesis is the conflict-driven miniSAT algorithm described in [3, 4]. The algorithm converts pseudo-boolean constraints and the objective functions into clauses and then sees if the problem is satisfiable. In the first iteration, they form clauses from the objective function such that it is assumed that its cost is smaller than a very large k value. In each iteration, they tighten or loosen the condition depending on whether the k value used results in a set of satisfiable or unsatisfiable clauses. This is done repeatedly until the optimum solution is found. If the problem is unsatisfiable in the first iteration, the problem can be quickly confirmed as unsatisfiable.

1.3.2 Traffic Modelling/Prediction

Since the formulation of the Fundamental Diagram by Bruce Greenshields in 1930s, work is still being done on studying and modelling of traffic speed, density and flow relationships and their uncertainties [5,7,9]. It is interesting to note that [34] presents such a study specific to the CBD area in Singapore. In figures 1.1, 1.2, 1.3 and 1.4 we can see some general trends discussed. In figure 1.1, we see that given the traffic density, the corresponding traffic

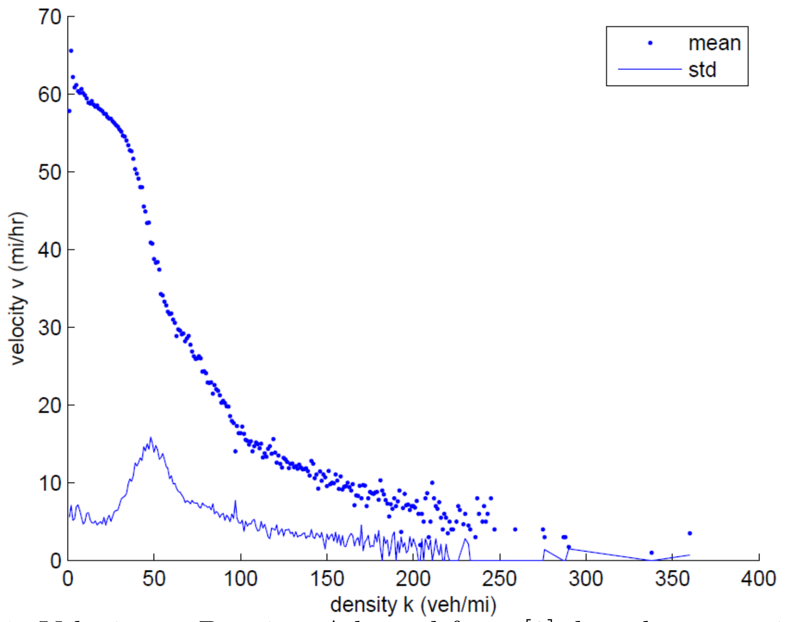


Figure 1.1: Velocity vs Density. Adapted from [9], based on a typical site of the GA400 ITS dataset.

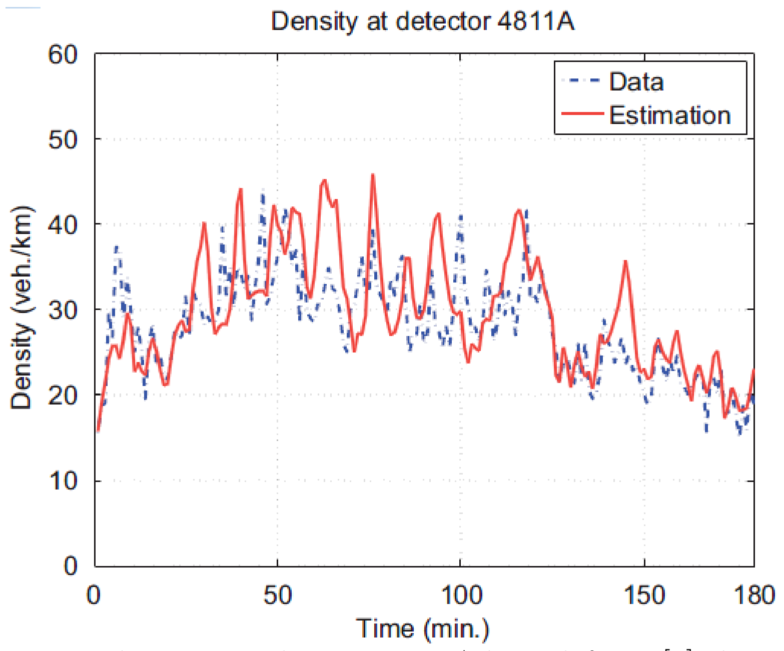


Figure 1.2: Road Density Fluctuation. Adapted from [7], based on data collected by detector 4811A along M25 freeway in England.

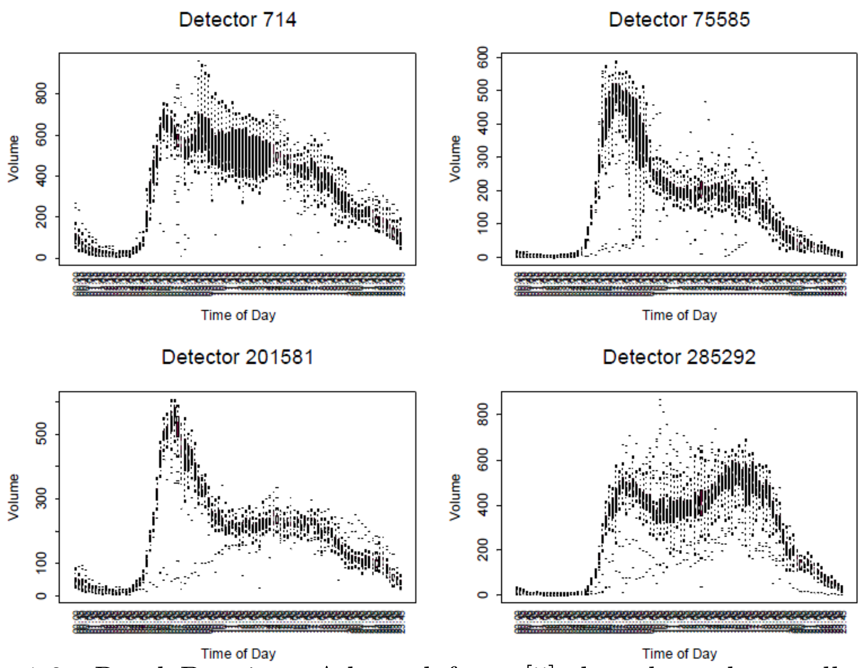


Figure 1.3: Road Density. Adapted from [5], based on data collected by various detectors along Atlanta interstate highway.

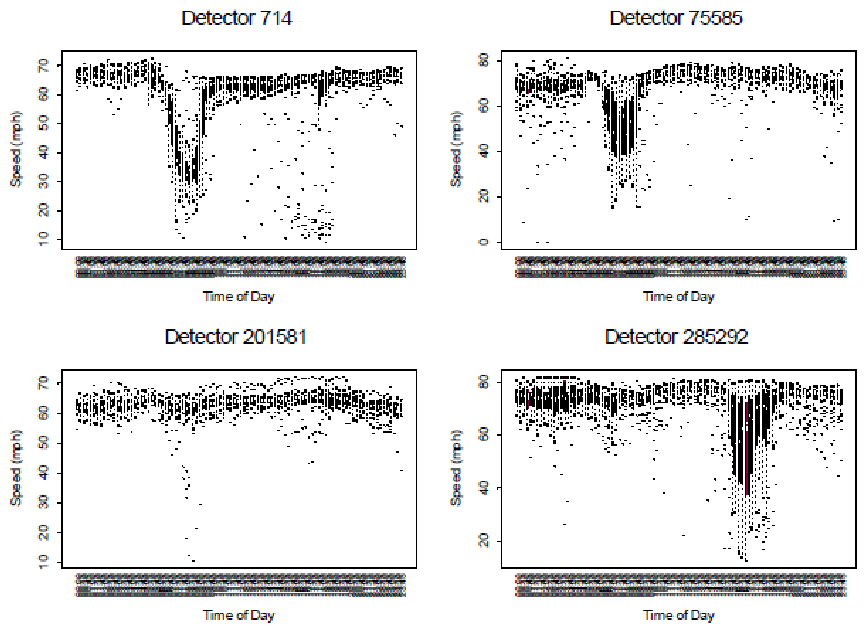


Figure 1.4: Roads Speed. Adapted from [5], basd on data collected by various detectors along Atlanta interstate highway.

speed is distributed probabilistically. Note that while the diagram represents a general trend, it is not always true. In figure 1.2, we see an example of how traffic density fluctuates at high frequencies but the values are typically bounded within a range at any time instance. In figure 1.3, we see a plot of traffic density in four different roads in Georgia against the time of the day throughout the year of 2003 [5]. The plot indicates that while different roads differ from each other a lot, they also follow a similar trend to some extent; the most common trend being the peak in traffic density in the morning. In addition, for each road and the given time of the day, there is a distribution of traffic density which could be modelled into a probability distribution.

Figures 1.4 and 1.3 shows the car density and traffic speed on the same 4 roads respectively. Here, we can see some examples contradictory to the Fundamental Diagram especially in the case for "Detector 201581" where the average traffic speed seems almost invariant with respect to the varying car density.

As a general guide to traffic speed according to [8], traffic speed is found to be distributed normally example from [8] shown in figure 1.5, i.e. they are Gaussian distributions. Therefore if we know the specific expectation and variance of each edge E_i at time t, they can be represented as $\mathcal{N}(\mu_{t,E_i}, \sigma^2_{t,E_i})$. Note that traffic speeds uses the units of km/hr even though the cited example uses the units of mph.

Note that while it is stated that t is the time of the day, the definition can be modified to have a better resolution depending on data availability, i.e. it is possible to also consider the day of the week, month of the year etc. This allows for better data analysis since day-to-day and seasonal situation have unique characteristics, i.e. significantly less people goes to work on the weekends, snowing in December, annual holidays etc. We can see work in [10] that studies this aspect of influence to traffic conditions. In addition, there exist holidays that depends on non-Gregorian Calendars and that can also be easily accounted for by considering these days separately. However, a reasonably large amount of independent samples are necessary for μ_{t,E_i} and σ_{t,E_i} to be reliable. Therefore, better resolutions in the definition of t is not always possible and it depends on the amount of past information available.

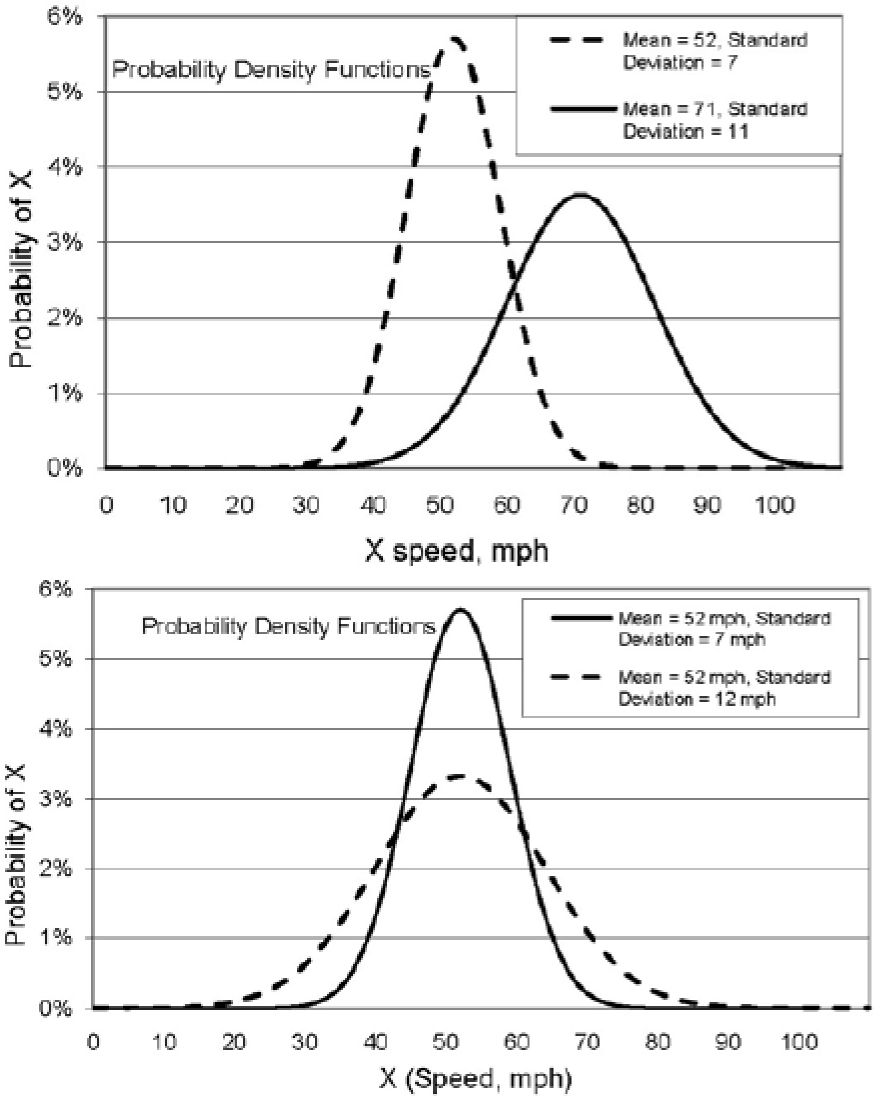


Figure 1.5: Speed Normally Distributed. Example plots adapted from [8].

1.3.3 City-Scale Traffic Forecasting with Roving Sensor Network

While the probe allocation problem is known to be novel, there exists one paper that tackles a similar problem. In [12], they estimate traffic car density by making inference based on the number of roving sensors (taxis) that are on each road. They compare their estimations with the ground truth indicated by information collected by loop detectors that are already installed on roads. The data used in [12] is identical to the ones used in this thesis and they are provided by LTA and ComfortDelgro. More details of the data can be found in section 5.1.

While the roving sensors (taxis) information was able to provide variations similar to that which is presented in the ground truth, there exists a form of bias that changes throughout the day. The bias is a constant deviation from the real data at each particular time of the day but varies throughout the day, i.e. taxi information can be higher than the ground truth by a certain consistent value at 8am for a particular road but is lower than the ground truth by another value at another time of the day perhaps 1pm and such deviation is found to be a phenomenon that is repeated each day. However, it is found that for the same time of each day the bias is consistent. Therefore, the bias could be learnt if enough past information is provided and if the bias is taken into account, accurate estimations can be made about the traffic density of each road. From the results presented by [12] it is found that one week of past information is sufficient to learn the bias for each road.

1.3.4 Bus Route Planning

The problem of bus route planning is a NP-hard problem and therefore any algorithm proposed to solve this problem is found to be stochastic in nature or at least involves heuristic methods [20–28]. This has been the common practise in period as early as 1974 [27]. The main contribution of this thesis is therefore a novel approximation algorithm that is deterministic and it is shown that the proposed algorithm is superior to many existing algorithms.

Each of the papers in [20–28] is unique in their own unique way. For example, [24] presents a novel ideal of modelling travel demand and distance of bus terminals in a form of Newton gravity. In [20], it has been considered that departure time for buses can be modified in order to minimize transfer time. In [21], users are allowed to input certain important design parameters such as the number of initial routes (skeletons), expansion strategy and identification of termini nodes. In [25], a metaheuristic search scheme that combines simulated annealing, tabu, and greedy search methods was introduced. In [27], it is assumed that passengers can take any route that serves their desired OD pair. Meaning, the bus waiting time is reduced as commuter route choices increases. In [28], the problem is considered in 3 difference levels and tackled in loops through iterations. In the first loop speed of the transportation is considered; buses, trains etc. In the second loop, the areas are split into smaller geographical areas. In the third loop they have the more detailed route designs. In addition, various methods were adopted, for example Ant Colony Optimization was used for the route modification, Genetic Algorithm was used to decide frequency and fleet size and Headway-Based Stochastic Multiple Route was used to evaluate performance of the design in iteration. Despite the many different algorithms that has been proposed in various papers, the benchmark most commonly used for comparison of performance is the Mandl's Benchmark ever since it was created by Mandl in 1979 [22].

1.4 Problem Formulation

In this section, the details of the probe allocation problem are presented. We begin with the general problem statement followed by the formulation for the traffic speed model, entropy of traffic speed and a more formal description of the objective.

1.4.1 Probe Allocation Problem Statement

Let $G = (V, E)$ be a directed graph denoting the road network in an urban area, with edges E denoting roads and vertices V denoting road junctions. Let τ_{E_i} be the average time it takes to travel along each edge E_i and transit to another edge, which is the result of the average travel speed along each edge.

Let $g(E_i, t)$ be the actual traffic speed along edge E_i at time t. The entropy/uncertainty of the traffic estimation for edge E_i at time t is $\tilde{g}(E_i, t)$, let $\tilde{g}^{\max}(E_i)$ be the highest value of entropy for edge E_i and let β_{t,E_i} be the rate at which the entropy increases for edge E_i at time t. Assuming that no measurement is made between time t and $t+\delta$, then the entropy at time $t+\delta$ is $\min(\tilde{g}(E_i,t)+\delta\beta_{t,E_i}, \tilde{g}^{\max}(E_i))$.

Each time a vehicle transits E_i, it can make a measurement $\hat{g}(E_i,t)$ of the quantity and resets $\tilde{g}(E_i,t)$ to a certain initial value. The objective is to design an algorithm to determine control input and number of agents required such that the entropy/uncertainty of the traffic estimation $\tilde{g}(E_i,t)$ are bounded, i.e. $\tilde{g}(E_i,t) \leq \epsilon \; \forall E_i, t$. In this problem, the vehicles are known as agents and they will measure the traffic speed of each road as they traverse the roads.

Assuming that in order to ensure $\tilde{g}(E_i,t) \leq \epsilon \; \forall E_i, t$, each road E_i will have to be visited by an agent at a certain frequency f_{t,E_i}. In chapter 2, we look at the special case where we assume that $f_{t,E_i} = f_{t,E_j} \forall t; E_i \neq E_j$. In chapter 3, we look at the general case where the assumption is removed. The two problems are presented in two separate chapters as their solutions differ significantly. In chapter 4, we look at the same problem as chapter 3. However, random walk solutions are looked at in chapter 4 with contrast to the fixed circuit solution in chapter 3. Finally in chapter 6, we will look at the bus route planning problem. Although the bus route planning problem is similar to the probe allocation problem, its problem statement will only be presented in chapter 6 to avoid confusion.

1.4.2 Traffic Speed Model

Despite previous work [8,9,30] having modelled traffic speed as a Gaussian distribution, the method still pose problems if it assigns a significant amount of probability to invalid speeds, i.e. -5 km/hr or 200 km/hr. In these cases, the approximation of traffic speed as a Gaussian distribution is invalid.

Let $\hat{s}_{E_i}$ and $\check{s}_{E_i}$ be the maximum and minimum valid speed on edge E_i respectively. The total probability between $\hat{s}_{E_i}$ and $\check{s}_{E_i}$ is:

$$\lambda_{t,E_i} = \int_{\check{s}_{E_i}}^{\hat{s}_{E_i}} \mathcal{N}(\mu_{t,E_i}, \sigma^2_{t,E_i}). \tag{1.1}$$

Let $p^s(s;t,E_i)$ be the probability function of speed on edge E_i given the time of the day, we now define a modified Gaussian distribution where the probability for invalid speeds is zero and total probably is still unity:

$$p^s(s;t,E_i) \begin{cases} = \frac{\mathcal{N}(\mu_{t,E_i},\sigma^2_{t,E_i})}{\lambda_{t,E_i}} & if \check{s}_{E_i} \leq s \leq \hat{s}_{E_i} \\ = 0 & otherwise \end{cases}. \tag{1.2}$$

Ever since the formulation of the Fundamental Diagram by Bruce Greenshields in 1930s, the Speed-Flow-Density relationship of any traffic speed model is something that will usually be of interest to any researcher. In this thesis, however, we are only interested in modelling the traffic speed and the study of the Speed-Flow-Density relationships will not be carried out. For those interested, here are some examples of work in this direction [5, 7, 9, 34]. It is interesting to note that [34] presents a study specific to the CBD area in Singapore.

1.4.3 Entropy of Traffic Speed

In information theory, entropy is a measure of the uncertainty associated with a random variable and we first recall the definition of continuous entropy in Nats is $-\int_{-\infty}^{\infty} f(x)\ln(f(x))$ [31–33]. Let $f(x) = p^s(s; t, E_i)$. We then define maximum entropy $\tilde{g}^{\max}(E_i, t)$ as the highest amount of entropy for edge E_i at time t based on past information, i.e. the amount of entropy for each edge cannot exceed $\tilde{g}^{\max}(E_i, t)$ even if no agent passes by the edge since past information itself can be used as an estimation of the traffic speed with an amount of entropy of its own. Meaning to say, even if no measurement was taken at all the past information can still be used to estimate the traffic speed. The maximum entropy is calculated to be:

$$\begin{aligned}
&\tilde{g}^{\max}(E_i,t) = -\int_{-\infty}^{\infty} f(x)\ln(f(x)) \\
&= -\int_{\check{s}_{E_i}}^{\hat{s}_{E_i}} f(x)\ln(f(x)) = -\int_{\check{s}_{E_i}}^{\hat{s}_{E_i}} \psi e^{-\xi}\ln(\psi e^{-\xi}) \\
&= \int_{\check{s}_{E_i}}^{\hat{s}_{E_i}} \psi e^{-\xi}\ln(\tfrac{1}{\psi}) + \int_{\check{s}_{E_i}}^{\hat{s}_{E_i}} \psi\xi e^{-\xi} \\
&= \ln(\tfrac{1}{\psi}) - \psi\left[\tfrac{s-\mu_{t,E_i}}{2}e^{-\xi}\right]_{\check{s}_{E_i}}^{\hat{s}_{E_i}} + \int_{\check{s}_{E_i}}^{\hat{s}_{E_i}} \tfrac{1}{2}\psi e^{-\xi} \\
&= \ln(\tfrac{1}{\psi}) - \psi\left[\tfrac{s-\mu_{t,E_i}}{2}e^{-\xi}\right]_{\check{s}_{E_i}}^{\hat{s}_{E_i}} + \tfrac{1}{2},
\end{aligned} \tag{1.3}$$

where $\psi = \frac{1}{\lambda_{t,E_i}\sigma_{t,E_i}\sqrt{2\pi}}$ and $\xi = \frac{(s-\mu_{t,E_i})^2}{2\sigma_{t,E_i}^2}$ [1].

Let $g(E_i, t)$ be the actual traffic speed along edge E_i at time t. Each time a vehicle transits E_i, it can make a measurement $\hat{g}(E_i, t)$ of the quantity. Let

[1]Integration by parts was used to reach line 4 of equation (1.3). Let the integration be $\int u(x)v'(x)dx = u(x)v(x) - \int u'(x)v(x)dx$ where $u(x) = \frac{s-\mu_{t,E_i}}{2}$ and $v'(x) = \frac{s-\mu_{t,E_i}}{\sigma_{t,E_i}^2}e^{-\xi}$ and $x = s - \mu_{t,E_i}$.

$\bar{\mu}_{t,E_i}$ and $\bar{\sigma}^2_{t,E_i}$ be the expectation and variance of our estimation of the traffic speed at time t on edge E_i. The initial entropy is then be given by:

$$\tilde{g}(E_i,t) = \ln(\tfrac{1}{\psi}) - \psi \left[\tfrac{s-\bar{\mu}_{t,E_i}}{2} e^{-\xi} \right]_{\check{s}_{E_i}}^{\hat{s}_{E_i}} + \tfrac{1}{2}, \tag{1.4}$$

where $\psi = \frac{1}{\lambda_{t,E_i}\bar{\sigma}_{t,E_i}\sqrt{2\pi}}$ and $\xi = \frac{(s-\bar{\mu}_{t,E_i})^2}{2\bar{\sigma}^2_{t,E_i}}$ [2].

Assuming that uncertainty in our estimations increase linearly with time at a rate β_{t,E_i}. Provided that no measurement is made between time t and $t+\delta$, then the entropy at time $t+\delta$ is [3]:

$$\tilde{g}(E_i, t+\delta) = \min(\tilde{g}(E_i,t) + \delta\beta_{t,E_i}, \tilde{g}^{\max}(E_i,t)). \tag{1.5}$$

1.4.4 Objective

The objective is to design an algorithm to determine control input and number of agents required such that the entropy/uncertainty of the traffic estimation $\tilde{g}(E_i,t)$ are bounded, i.e. $\tilde{g}(E_i,t) \leq \epsilon \; \forall E_i, t$ [4]. As a simplification to the problem, we consider a quasi-static problem where $\beta_{t,E_i}, \mu_{t,E_i}, \sigma_{t,E_i}$ and $\tilde{g}^{\max}(E_i,t)$ remains the same from time t (when the last measurement was taken) to time $t+\delta$ (where the next measurement should be taken). Therefore the frequency at which each edge E_i has to be visited by an agent is:

$$f_{t,E_i} = \begin{cases} 0 & \text{if } \epsilon \geq \tilde{g}^{\max}(E_i,t) \\ \frac{1}{\delta} = \frac{\beta_{t,E_i}}{\epsilon - \tilde{g}_(E_i,t)} & \text{otherwise} \end{cases}. \tag{1.6}$$

1.4.5 Simulation Setup (NUS)

As chapters 2 to 4 shares the same NUS simulation scenario with slight variations, an overview of the NUS simulation will be shown here to give a general idea of the NUS simulations. Explanation of the details of simulation in each chapter is provided within the respective chapters.

[2]When $\tilde{g}(E_i,t) = \tilde{g}^{\max}(E_i,t)$, $\bar{\mu}_{t,E_i} = \mu_{t,E_i}$ and $\bar{\sigma}_{t,E_i} = \sigma_{t,E_i}$.

[3]Since higher fluctuation naturally indicate faster rise in uncertainty, one way to determine β_{t,E_i} is to measure the noise in past information on edge E_i at time t.

[4]Note that if a new measurement is taken at time t on all edges, then $\epsilon > \max(\tilde{g}(E_i,t)) \forall E_i$ otherwise the problem is unsolvable.

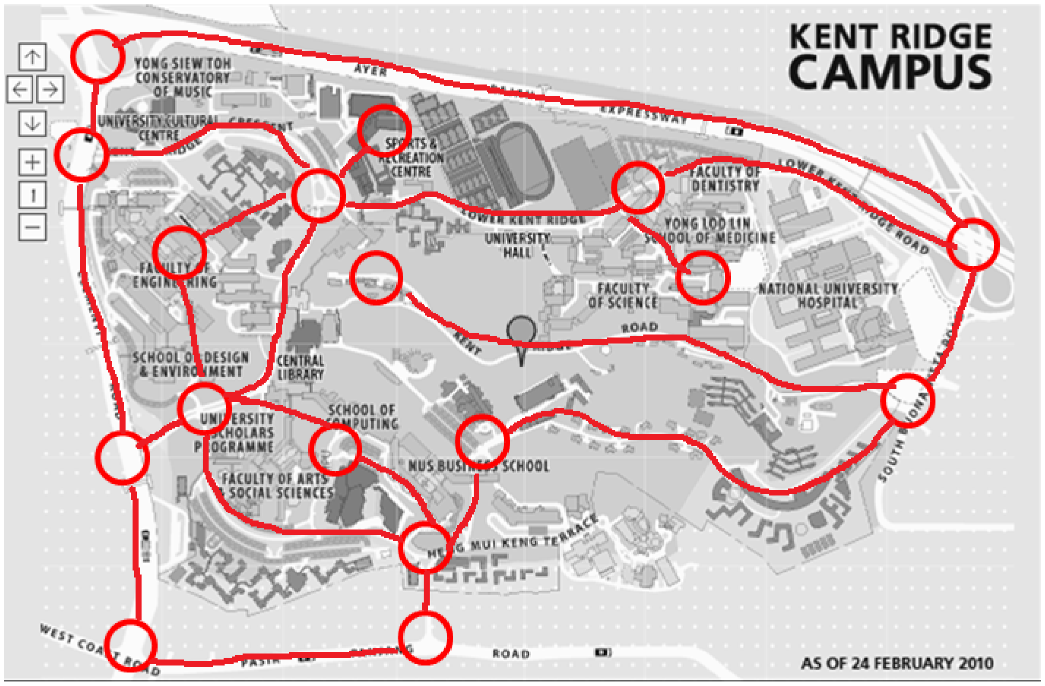


Figure 1.6: Original NUS Map.

Although the NUS simulation involves a scenario created based on personal experiences, it is setup such that the scenario is as believable as possible. The setting is that of a morning peak hour where it require more agents than other hours of the day. In figure 1.6 we can see the original NUS map the simulation is based on. In figure 1.7, we can see the graph drawn from the map of NUS with which the simulation is carried out. Along each edge, next to each arrow head is a number representing the amount of time in minutes it takes to travel along the edge in the direction pointed by the arrow head. There is also a number on the middle of each edge representing the period at which each edge should be visited in minutes.

The values used in figure 1.7 are estimated based on personal experience but are believed to be reasonable estimation due to my personal experience travelling in NUS while being a student in the university; I have travelled along each of these roads at least once either on a bus or on a car etc. For example, in the morning peak hour, it will be faster to travel from Node 1 to Node 11 than the opposite direction. The periods at which each edge should be visited are also estimated based on my personal believe of how

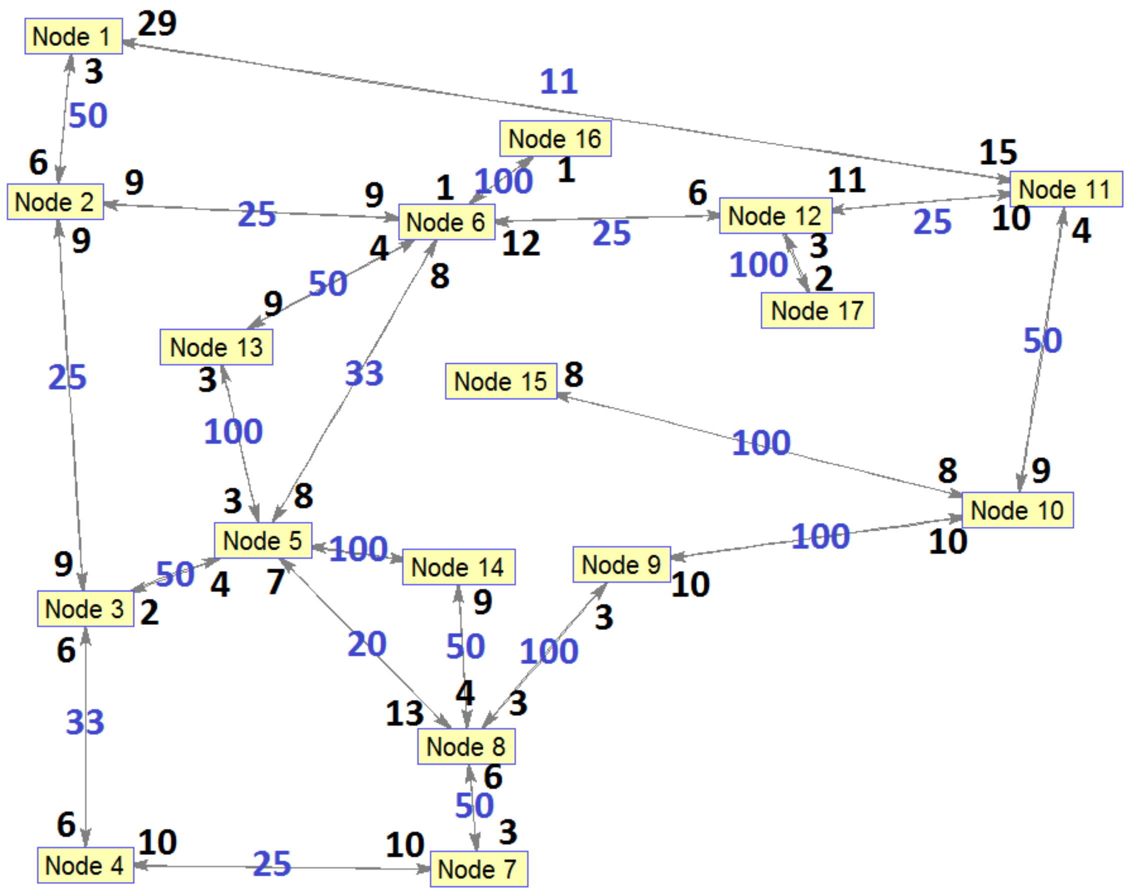


Figure 1.7: NUS Simulation Environment. Along each edge, next to each arrow head is a number representing the amount of time in minutes it takes to travel along the edge in the direction pointed by the arrow head. There is also a number on the middle of each edge representing the period at which each edge should be visited in minutes.

predictable each road's traffic condition are. For example, the road between Node 10 and Node 15 is usually always empty and does not need to be visited frequently while the road between Node 1 and Node 11 is a highway and its traffic condition may vary a lot over a small amount of time and therefore it was believed that it should be visited once every 11 minutes.

1.4.6 Summary

In this chapter, we presented the details of the probe allocation problem. A modified Gaussian distribution was made and used as a model of the traffic speed distribution of each road. We derived the closed-form expression for the entropy of the traffic speed distribution and its variation with time. Then we calculate the frequency at which each edge has to be visited. With this information, we were able to begin introducing algorithms as solutions to solve the probe allocation problem.

Chapter 2

Relaxed Eulerian Circuit Solution

The probe allocation problem described in section 1.4 is generally a NP-hard problem. However, there exists a special case of the probe allocation problem such that all edges are required to be visited at the same frequency, i.e. $f_{E_i} = f_{E_j} \forall E_i, E_j \in E$. This special case of the probe allocation problem can be solved in $\mathrm{O}(n^3)$ computational steps. Such situation arises when no prior information is available on the traffic statistics of each road and therefore it is reasonable to assume that the growth rate of each road is the same.

In order to solve this problem, it is necessary to find the shortest circuit of the given graph G such that it visits all edges at least once. This is proven in sections 2.2. We define the *relaxed eulerian circuit* as the shortest possible circuit in G that uses every edge at least once. The term "relaxed" is used since edges are allowed to be visited more than once. Eulerian Circuit has a similar definition but edges are visited exactly once.

Define $D^{+/-}(V_j)$ as the result from the difference between the in-degree and out-degree of node V_j, i.e. if in-degree of V_j is 2 and out-degree is 4 then $D^{+/-}(V_j) = 2 - 4 = -2$. Essentially, to find the relaxed eulerian circuit we need to balance the in-degree and out-degree of all nodes in G, i.e. $D^{+/-}(V_j) = 0 \forall V_j$. If a graph G has the property $D^{+/-}(V_j) = 0 \forall V_j$, finding its Eulerian Circuit can be quite simply done. This is shown in section 2.4.3.

When there exits some V_j such that $D^{+/-}(V_j) \neq 0$, it is necessary to introduce pseudo-edges so that $D^{+/-}(V_j) = 0$. Note that

$$\sum_{\forall V_j} D^{+/-}(V_j) = 0 \tag{2.1}$$

is always true since each edge contributes exactly one in-degree and one out-degree in graph G. Therefore, imbalanced degree in nodes V_j^+ such that $D^{+/-}(V_j) > 0$ can be cancelled out with imbalance degree in nodes V_j^- such that $D^{+/-}(V_j) < 0$. The strategy is to compute the shortest path for all V_j^+ to all V_j^- in G and pair up nodes of different sets so that we repeat edges along their shortest path. We introduce pseudo-edges into G for these edges along the shortest paths so that $D^{+/-}(V_j) = 0 \forall V_j$. However, to ensure optimality of the solution it is necessary to pair nodes such that the sum of costs of repeated edges is minimized.

Assuming that the relaxed eulerian circuit is found, let τ_{C} be the time it takes to travel one cycle of the circuit. The number of agents N required is $\min(N) : N/\tau_{\mathrm{C}} \geq f_{E_i}$ for any edge E_i. The agents are sent to traverse the circuit repeatedly such that each of them are separated by a time difference of τ_{C}/N away from its immediate neighbours along the circuit. Therefore, all points in the circuit are visited by an agent once every τ_{C}/N time has passed and therefore are visited by an agent at a frequency of at least N/τ_{C} [5].

2.1 Chinese Postman Problem

A similar problem was studied in 1962 by a Chinese mathematician Mei-Ku Kuan and the problem is known as the Chinese Postman Problem [1,29]. The problem in essence is to find the shortest possible closed circuit or path that visits every edge at least once in an undirected graph. The different between the relaxed eulerian circuit problem and the Chinese Postman Problem is that relaxed eulerian circuit is defined for directed graphs while Chinese Postman Problem is defined for undirected graphs.

Despite the difference between the two problems, they are essentially very similar problems and can be solved using similar solutions. Both problems require us to perform a node pairing optimization to facilitate the search for the relaxed eulerian circuit. In the case of Chinese Postman Problem the node pairing is performed on odd nodes so that all nodes are of even degree.

[5] The points along edges that are repeated is visited at a higher frequency.

2.2 Optimality Of Relaxed Eulerian Circuit

In this section, proof is provided to show the relaxed eulerian circuit gives optimum solution for the special case of the probe allocation problem stated at the start chapter 2. Recall that τ_{C} represents the time taken to traverse one cycle in the relaxed eulerian circuit.

Theorem 1. The relaxed eulerian circuit is the optimum circuit assignment for any surveillance problem where all edges are required to be visited by agents at the same frequency, i.e. $f_{E_i} = f_{E_j} \forall E_i, E_j \in E$.

Proof:
Case 1. Single circuit solution.

First note that the relaxed eulerian circuit is the shortest possible circuit in G that uses every edge at least once by definition. Therefore, there exists no other circuit in graph G that is shorter [6].

Case 2. Multiple circuit solution.

Assuming that there is k number of circuits and each of them requires $\tau_{\mathrm{C},i} \forall i = \{1 \dots k\}$ amount of time to traverse one cycle. The number of agents required by the relaxed eulerian circuit is given by $\lceil f_{E_i} \tau_{\mathrm{C}} \rceil$ while the number of agents required by the multiple circuit is given by

$$\sum_{\forall i=\{1\dots k\}} \lceil f_{E_i} \tau_{\mathrm{C},i} \rceil . \tag{2.2}$$

However, as we have multiple circuits we know that

$$\tau_{\mathrm{C}} \leq \sum_{\forall i=\{1\dots k\}} \tau_{\mathrm{C},i} \tag{2.3}$$

since having multiple circuits may mean the need to repeat additional edges or a suboptimum selection of edges. Therefore since any f_{E_i} are of the same value,

$$f_{E_i} \tau_{\mathrm{C}} \leq \sum_{\forall i=\{1\dots k\}} f_{E_i} \tau_{\mathrm{C},i} . \tag{2.4}$$

[6] The reasoning here is very self-explanatory and is the same reason why the Chinese Postman Problem is solved in a similar way.

Since lesser ceiling operations means lesser amount of rounding up,

$$\lceil f_{E_i}\tau_{\mathrm{C}} \rceil \leq \sum_{\forall i=\{1\ldots k\}} \lceil f_{E_i}\tau_{\mathrm{C},i} \rceil . \tag{2.5}$$

Therefore, having multiple circuits does not reduce the number of agents required. This implies that the solution relaxed eulerian circuit is optimum since it is optimum in Case 1 and solutions from Case 2 cannot perform better than it. □

2.3 Node Pairing: Graph Theory Vs. Combinational Optimization

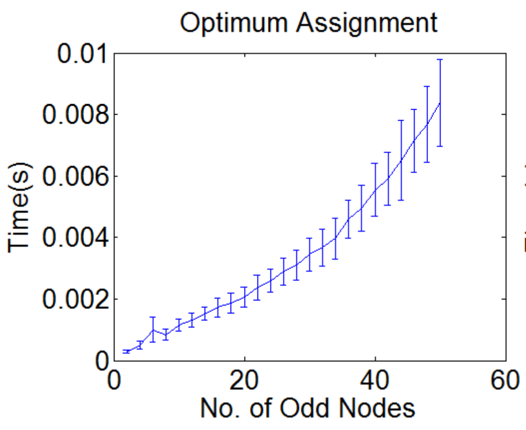
As both the relaxed eulerian circuit problem and the Chinese Postman Problem requires the process of pairing up nodes, in this section a comparison between two existing algorithms is carried out. The first algorithm is the Kuhn-Munkres Algorithm or the *Optimum Assignment Algorithm*, and it is described in [1]. It is known that this algorithm has a computational complexity of O(n^3). It is noteworthy that there exists another similar algorithm called the Hungarian Algorithm [14]. It shares the same complexity and can be used to solve the node pairing problem as well.

The Optimum Assignment Algorithm as described in [1] essentially finds the optimum solution by looking for a perfect matching in a graph through iterations. A matching is a set of edges in a graph without common vertex, while a perfect matching is a matching in a graph such that all vertex of the graph is matched. Given any weighted bipartite graph G, a spanning subgraph of G, G_f is created such that there are lesser edges in G_f than G. G_f have lesser edges as some of them are excluded based on a constraint defined in the algorithm. The algorithm then attempt to find a perfect matching in G_f, the problem is solved if there is a perfect matching. If a perfect matching was not found, the constraint defined by the algorithm is relaxed. G_f is then updated based on the relaxed constraint so that it has more edges in it. Starting from the matching found in the previous iteration, it continues the attempt to find a perfect matching in G_f. The whole process is repeated until a perfect matching is found.

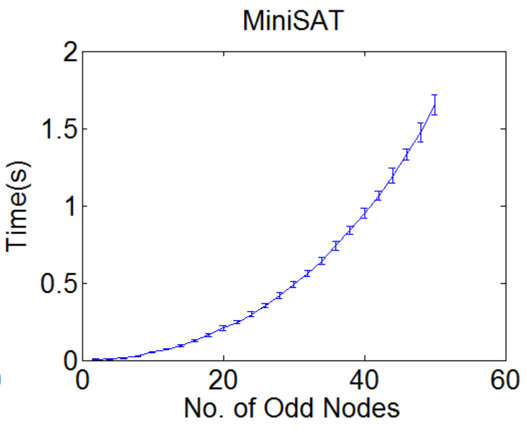
The second algorithm is the miniSAT algorithm [3, 4] that have been awarded for its performance in the SAT 2005 competition. It is known that

SATsolvers tend to perform well in Boolean combinatorial constrained optimization problems [6] not to mention the attention given to SATsolvers in the yearly competitions.

The comparison is done on 1,250 randomly generated node pairing problems. The node pairing problems range from having only 2 nodes to having 50 nodes, as even number of nodes are required therefore there are 25 variations of problem size within that range. For each problem size, 50 problems are randomly generated and therefore there are a total of 1,250 problems. In each randomly generated problem the all nodes can be paired with any other nodes except for itself and the cost is randomly generated.



(a) Computational Time For Optimum Assignment



(b) Computational Time For MiniSat

Figure 2.1: Node Pairing Computational Time

As we can see in figure 2.1, the Optimum Assignment Algorithm is much faster than the miniSAT algorithm. This is expected since the Optimum Assignment Algorithm is an algorithm written to specifically solve problems such as the node pairing problem. However SATsolvers in general are still very useful in solving general Boolean combinatorial constrained optimization problems. In each plot in figures 2.1a and 2.1b, the line is drawn across the mean computational time of each problem size. The bar at each problem size indicates the maximum and minimum computational time.

2.4 Relaxed Eulerian Circuit Algorithm

Two solutions are introduced in this section; the first is a modified solution to the Chinese Postman Problem which is applicable to undirected graphs. The modification made is a pre-processing to remove all degree-1 nodes from being considered for node pairing. This process is of O(n) complexity and reduces the number of nodes considered for node pairing which is a O(n^3) process. The second is the relaxed eulerian circuit Algorithm which is applicable to directed graphs. The complexity of solutions for either case whether the graph is undirected or directed is O(n^3). n here correspond to the number of nodes on either side of the bipartite graph that is constructed before applying the algorithm. The bipartite graphs are described later in each subsection.

2.4.1 Modified Chinese Postman Algorithm (Undirected Graph)

While there is a polynomial time solution for the Chinese Postman Problem, it is still helpful to identify ways to improve the computational time. Here, we identify the fact that degree-1 nodes are odd nodes that contributes to the complexity of the solution in the O(n^3) node pairing process. However, it is obvious that the edge incident to it is repeated and added as a pseudo-edge since it is the only possible option.

Let E'_{CP} and E''_{CP} be a list of edges. Let the function $D(V_i)$ represent the degree of vertex V_i and let e^1 represent the set of edges incident to $V_i : D(V_i) = 1 \forall V_i \in V$ and let $d_s(V_i, V_j)$ be the time taken to travel along the shortest from vertex V_i to V_j and $e_s(V_i, V_j)$ is the set of corresponding edges.

```
   input: $G$,$E$,$V$
   output: $G$,$E$,$V$
1: Initialize $E'_{\mathrm{CP}} \leftarrow \emptyset$
2: **while** $\exists V_i \in V : D(V_i) = 1$ **do**
3:     Update $E'_{\mathrm{CP}} \leftarrow E'_{\mathrm{CP}} \bigcup e^1$
4:     Update $E \leftarrow E \setminus e^1$
5: **end while**
6: Update $G \leftarrow (V, E)$
7: Construct a bipartite graph $G'$ with bipartition as follows:
    $X = \{x_1, x_2, \ldots\}$ = odd vertices in$V$
    $Y = \{y_1, y_2, \ldots\}$ = odd vertices in$V$
    $w(x_i, y_i) = \infty$
    $w(x_i, y_j) = d_s(x_i, y_j) \forall i \neq j$
 8: Apply the Optimum Assignment Algorithm [1] on the bipartite graph $G'$
 9: Initialize $E''_{\mathrm{CP}} \leftarrow \emptyset$
10: for each node pair found in line 8 with a corresponding set of edges
    $e_s(V_i, V_j)$ do
11:     $E''_{\mathrm{CP}} \leftarrow \{E''_{\mathrm{CP}}, e_s(V_i, V_j)\}$
12: end for
13: Update $E \leftarrow \{E, E''_{\mathrm{CP}}\}$
14: Update $E \leftarrow \{E, E'_{\mathrm{CP}}, E'_{\mathrm{CP}}\}$
15: Update $G \leftarrow (V, E)$
```

The while loop from line 2 to 5 continuously remove edges from vertices that are degree-1 nodes, we do this repeatedly as this process may create new degree-1 nodes. The degree-1 nodes are now degree-0 since the edges incident to them are removed. In line 14 we not only add back the removed edges we also include them a second time as pseudo-edges as well.

In line 7, we construct a bipartite graph G' where X and Y are the two bipartitions with each nodes representing the odd vertices in V and the costs of the weights of edges corresponds to the shortest path distance between each node. However, since we cannot pair nodes with themselves the weight of x_i and y_i which corresponds to the same odd nodes are assigned infinite weight. In line 8 we apply the Optimum Assignment algorithm and carry out the node pairing process. In lines 9 to 12, we form a set of edges corresponding to the edges of the shortest path selected by the Optimum Assignment algorithm in line 8. These edges are then added into E as pseudo-edges.

2.4.2 Relaxed Eulerian Circuit Algorithm (Directed Graph)

In the case of a directed graph, most of the procedure is the same. However, it is important to ensure that the graph G is strongly connected. This procedure was found to be necessary while doing simulations with the Singapore road network provided by the LTA. The road network provided has source nodes with no in-coming edges and sink nodes with no out-going edges.

These nodes represents road in Singapore that are still under construction or are only partially constructed. It was found that there are 104 such nodes in Singapore. Agents will never visit source nodes and agents can never leave sink nodes once they enter, therefore removal of source and sink nodes is necessary otherwise the problem will have no feasible solution.

Let $e_i(V_i)$ represent the edges incident to node V_i. Define the function $D(V_i)_{in}$ and $D(V_i)_{out}$ as the number of in-coming edges and out-going edges incident to node V_i respectively. Let the function $R(V_i, k)$ be a set of repeated vertices V_i depending on the number k, i.e. $R(V_i, 5) = \{V_i, V_i, V_i, V_i, V_i\}$.

```
    input: $G$,$E$,$V$
    output: $G$,$E$,$V$
 1: while $\exists V_i \in V : D(V_i)_{in} = 0$ or $D(V_i)_{out} = 0$ do
 2:     Update $E \leftarrow E \setminus e_i(V_i)\ \forall\ V_i : D(V_i)_{in} = 0$ or $D(V_i)_{out} = 0$
 3:     Update $V \leftarrow V \setminus V_i\ \forall\ V_i : D(V_i)_{in} = 0$ or $D(V_i)_{out} = 0$
 4: end while
 5: Update $G \leftarrow (V, E)$
 6: Find $V^+ = \{R(V_i, |D^{+/-}(V_i)|)\}\ \forall\ V_i : D^{+/-}(V_i) > 0$
 7: Find $V^- = \{R(V_i, |D^{+/-}(V_i)|)\}\ \forall\ V_i : D^{+/-}(V_i) < 0$
 8: Construct a bipartite graph $G'$ with bipartition as follows:
    $X = \{x_1, x_2, \ldots\} = V^+$
    $Y = \{y_1, y_2, \ldots\} = V^-$
    $w(x_i, y_j) = d_s(x_i, y_j) \forall i, j$
 9: Apply the Optimum Assignment Algorithm [1] on the bipartite graph $G'$
10: Initialize $E''_{\mathrm{CP}} \leftarrow \emptyset$
11: for each node pair found in line 8 with a corresponding set of edges
    $e_s(V_i, V_j)$ do
12:     $E''_{\mathrm{CP}} \leftarrow \{E''_{\mathrm{CP}}, e_s(V_i, V_j)\}$
13: end for
14: Update $E \leftarrow \{E, E''_{\mathrm{CP}}\}$
15: Update $G \leftarrow (V, E)$
```

The while loop from lines 1 to 4 removes all source and sink nodes, the reason this is done in a while loop is because the process of removing source/sink nodes may produce new ones. In lines 6 and 7, we form the link of V^+ and V^- for pairing. Each nodes V_i are repeated with respect to the absolute value $|D^{+/-}(V_i)|$ as it determines the number of times each node V_i needs to be paired so that we can balance its in and out degrees, i.e. $D^{+/-}(V_i) = 0$. In lines 10 to 13 we form a set of edges corresponding to the edges of the shortest path selected by the Optimum Assignment algorithm in line 9. These edges are then added into E as pseudo-edges.

2.4.3 Hierholzer's algorithm

After either applying the modified Chinese Postman Algorithm or the relaxed eulerian circuit Algorithm depending on whether the graph is undirected or directed, we only need to apply Hierholzer's algorithm [15] to the graph to find the actual circuit agents would travel on, i.e. the sequence of nodes/edge. The algorithm has a complexity of $O(|E|)$ where $|E|$ is the number of edges including the pseudo-edges added. The algorithm first traverse the graph G at random until it reaches a vertex not incident to edges or pseudo-edges that have not been visited yet. The path (sequence of nodes/edges) so far will be stored and then the algorithm searches for vertices that is incident to any edges or pseudo-edges that have not been visited yet. From the vertex found the algorithm branches out to find a new path by continuing to traverse the graph from there (selecting only edges that have not been visited yet) until it reaches a vertex not incident to edges or pseudo-edges that have not been visited yet. The new branching path found is then inserted into the original path such that the new path formed is continuous. We repeat the process of finding a vertex incident to any edges or pseudo-edges that have not been visited yet and traversing the graph from there using only edges not yet visited until such vertex do not exist then the algorithm terminates. At the end of the algorithm, we would have a sequence of nodes/edges such that it is a circuit. More details can be found in [15].

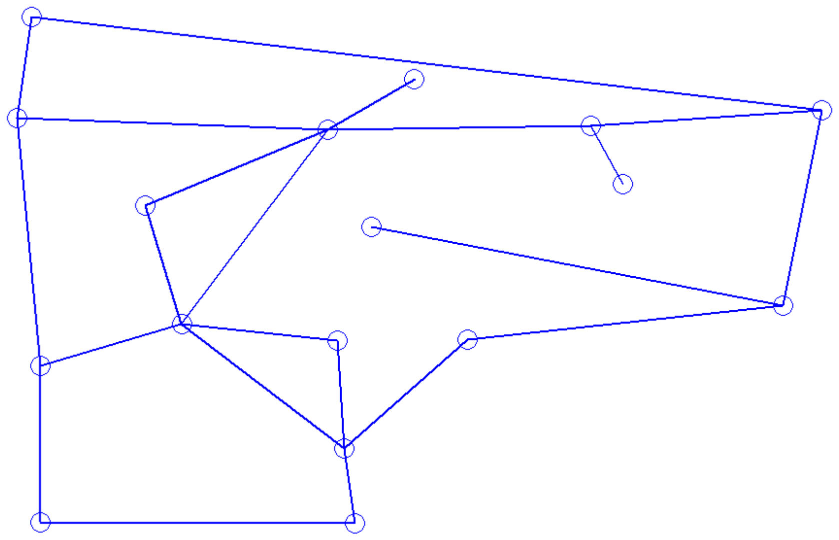
2.5 Simulation Results (NUS)

In the NUS simulation, the modified Chinese Postman Algorithm is used as all edges are assumed to be undirected. This is reasonable since each road in NUS has another road next to it that runs in the opposite direction. In addition, it is assumed that sensors on agents can measure traffic speed of roads on either direction. This assumption is only for this simulation. The road network with which the simulation is carried out can be seen in figure 1.6.

In figures 2.2 and 2.3 we can see the simulation results using the modified Chinese Postman Algorithm described in section 2.4.1. The details of the scenario for the simulation can be found in section 1.4.5. In figure 2.2a, we see a digitally drawn NUS road network based on the original NUS map in figure 1.6. In figure 2.2b, we see that the edges of degree-1 nodes are removed. In figure 2.3a, we see that four odd nodes are identified with nodes drawn as triangles and the edges that need to be repeated as pseudo-edges are identified. The selected node pairing is identified to be optimum and these edges indicated in dots-and-lines correspond to the shortest path between the odd node pairs. In figure 2.3b, we see that the edges of degree-1 nodes are added back twice and therefore are indicated with dots-and-lines to indicate that these edges are to be traversed twice (once for the original edge a second time for the pseudo-edge added). All other edges are to be traversed exactly once. Figures 2.2a, 2.2b, 2.3a and 2.3b clearly illustrate how the modified Chinese Postman Algorithm described in section 2.4.1 works.

2.6 Summary

In this chapter, we discussed the solution for the special case of the probe allocation problem. There were two versions of the problem and an algorithm was proposed for each of them, the first was when the graph provided was undirected and the second was when the graph provided was directed. Both cases involved the process of node pairing and it was found that Optimum Assignment Algorithm was faster in computing the optimum solution than a generic optimization solver as shown in section 2.3.


(a) NUS Road Network (Digitally Drawn)

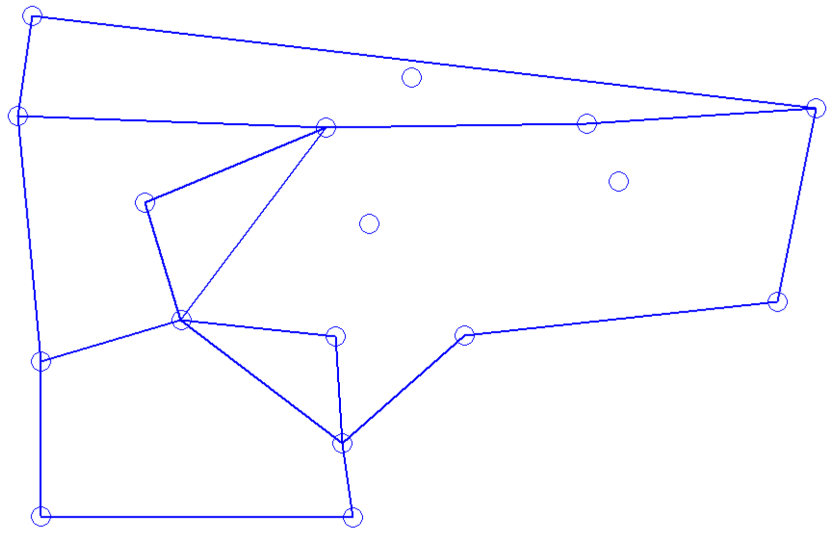
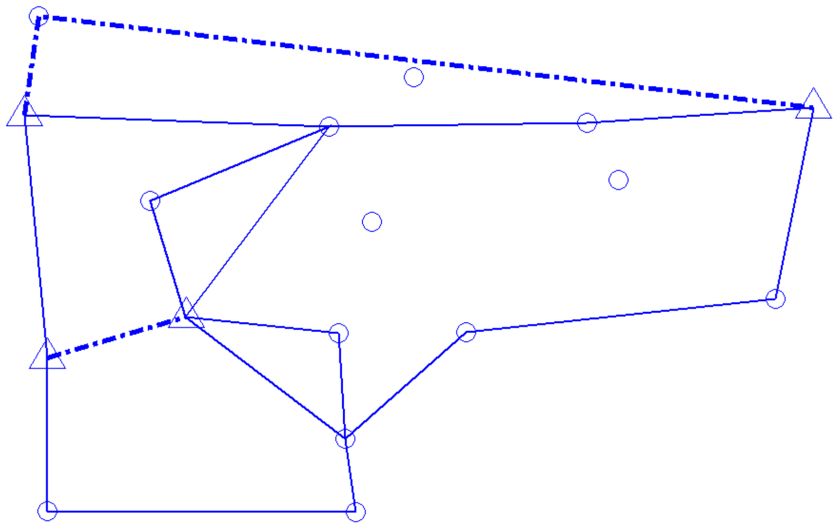
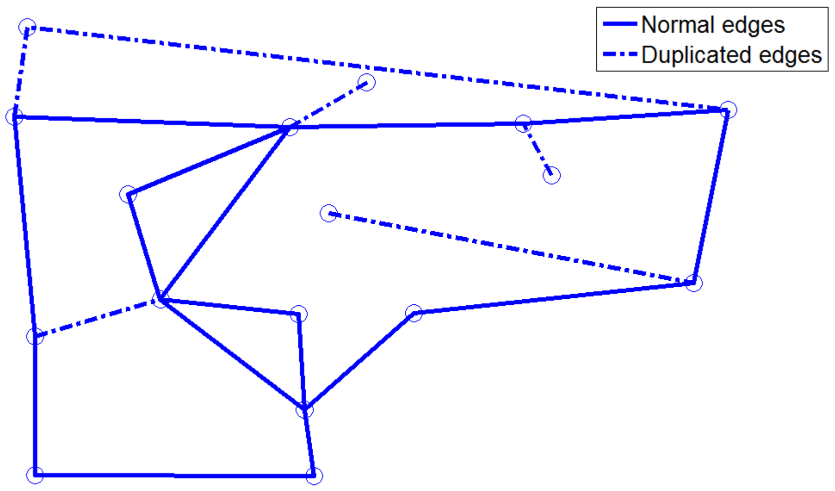

(b) Removal Of Degree-1 Nodes' Edges

Figure 2.2: Modified Chinese Postman Algorithm Simulation



(a) Node Pairing For NUS Road Network



(b) Resulting Modified Graph For NUS Road Network (Modified Chinese Postman Algorithm)

Figure 2.3: Modified Chinese Postman Algorithm Simulation

Chapter 3

Fixed Circuit Solution

In this chapter, we propose two algorithms each of which produces circuits for agents to traverse in as the solution for the stated problem. The solution is called fixed circuit solution as agents are expected to traverse the same circuit repeatedly [7].

In the two algorithm presented, the first is an algorithm that solves the problem by considering two basic desirable properties described in section 3.1, while the second is simply an approximation based on similar principles that has a even lower complexity.

It has been assumed that either the modified Chinese postman algorithm or the relaxed eulerian circuit Algorithm (depending on whether the graph G is undirected or directed) is applied to ensure that the problem is feasible, i.e. the fixed circuit solution may not be able to find the circuits in graph G if the required pseudo-edges are not added.

3.1 Problem Properties

The main objective of this section is to give an intuitive understanding of the solution proposed. The nature of this problem is similar to a surveillance problem such that each E_i has a given minimum frequency f_{Ei} at which it has to be visited by an agent. Due to the periodic nature of the problem, all agents are therefore assigned a particular circuit where they repeatedly

[7]In addition, the agents are spread out in their respective circuit evenly with respect to the circuit's total travel time, i.e. they are evenly spread out with respect to time instead of distance.

traverse.

A natural way to solve this problem is that we form k number of circuits α_k each consist of a subset of edges in E such that they are each being served by N_k number of agents at a frequency:

$$f_k^{\max} = N_k / \sum_{\forall E_i \in \alpha_k} \tau_{E_i}. \tag{3.1}$$

The objective is to minimize:

$$\sum_{\forall k} N_k, \tag{3.2}$$

subject to:

$$f_{E_i} \leq f_{k'}^{\max} \, \exists \, k' : E_i \in \alpha_{k'}. \tag{3.3}$$

However, the complexity of this problem formulation is far too high. Let each circuit k form a subgraph $G' = (V', E')$ of G where $V' \subseteq V$ and $E' \subseteq E$. Firstly there are $2^{|E|}$ ways to form each circuit k. Secondly, to check if each circuit is a single connected component requires a check on $\binom{|V'|}{2}$ conditions with a upper bound of $\binom{|V|}{2}$. Therefore, the complexity has a upper bound of $O(\binom{|V|}{2} 2^{k|E|})$.

In order to simplify the problem, we recognise that the optimum solution to the problem is the trade-off between two main properties:

1. Circuits consist of edges that are of similar frequencies.

2. Circuits have as few overlapping edges as possible with each other.

In the proposed solution, we break down the large original problem into a number of smaller problems where the circuits are optimized based on these two properties.

3.2 Original Fixed Circuit Solution

The original solution consists of two main parts. In the first part, each edge is clustered into different groups with respect to their f_{E_i} values. Then we solve for disjoint circuit α_k for each of these groups. Each disjoint circuit α_k is a subset of edges in E and forms a subgraph of G consisting of one or more components, and that there exists an Eulerian circuit in each of these components. In the second part, components are combined to form larger circuits to reduce number of agents needed. For example, if two circuits both need 0.3 agents, it is more efficient to combine them than to consider them separately because when combined they require only 1 agent but when considered separately they require a total of 2 agents.

As edges are clustered into groups such that edges of the same group have f_{E_i} values within a certain range. The solution found improves as the f range of each category decreases, i.e. when $k_{\max}$ increases where $k_{\max}$ is the maximum possible number of groups [8]. Eventually when all edges in each category consists of the same f_{E_i} value, the solution is always optimum, i.e. when $k_{\max} = k_{\text{opt}}$ [9] .

3.2.1 Combinatorial Optimization

The first part of the solution involves solving for the disjoint circuits for each group after having clustered the edges. This involves a combinatorial optimization which can be solved as a combinatorial optimization. The solver that is used for the simulation is the miniSAT solver [3, 4].

Assuming that the frequency values of each edge f_{E_i} and the total number of groups $k_{\max}$ are given. Let γ_k be the collection of all edges belonging to the k^{th} group and α'_k be a subset of γ_k consisting all edges that must be present in the disjoint circuit α_k. Define $f^{\min}$ as the non-zero minimum of f_{E_i} and $f^{\max}$ as the maximum of f_{E_i}:

$$f^{\min} = \min(f_{E_i}) \mid \forall E_i \in E : f_{E_i} \neq 0, \tag{3.4}$$

$$f^{\max} = \max(f_{E_i}) \mid \forall E_i \in E. \tag{3.5}$$

We calculate the frequency interval:

$$f' = \tfrac{f^{\max} - f^{\min}}{k_{\max}}. \tag{3.6}$$

Define the function $D(\alpha_k, V_j)_{in}$ and $D(\alpha_k, V_j)_{out}$ as the number of incoming edges and out-going edges in α_k for each vertex $V_j \in V$. The solution can be described in the following steps.

[8]Provided that edge clustering is affected, solution is the same if clustering is unchanged. The clustering can remain unchanged as we increase the number of groups as the groups can be empty if no edge has a f_{E_i} value that lies within the group's range.

[9]Since solution is optimum when $k_{\max} = k_{\text{opt}}$, increasing $k_{\max}$ beyond k_{opt} cannot improve solution.

1. Classify each edge E_i into the k^{th} group γ_k [10] :

$$\begin{aligned} &E_i \subseteq \gamma_k \mid k = \min\left(k' \,,\, k_{\max}\right); \\ &k' : k' - 1 \leq \tfrac{f_{E_i} - f^{\min}}{f'} < k'. \end{aligned} \tag{3.7}$$

2. Define max frequency of k^{th} group as:

$$f_k^{\max} = \max(f_{E_i})\,\forall\, E_i \in \gamma_k. \tag{3.8}$$

   Construct the objective function:

$$\sum_{\forall k \in [1, k_{\max}]} \left(f_k^{max\,2} \sum_{\forall E_i \in \alpha_k} \tau_{E_i} \right). \tag{3.9}$$

3. Construct the following constraints for each $k \in [1, k_{\max}]$:

   (a) $\alpha'_k \subseteq \gamma_k$, i.e. α'_k is the subset of edges that belongs to the k^{th} group.

   (b) $\alpha_k \subseteq E$, i.e. α_k is the subset of all available edges,

   (c) $D(\alpha_k, V_j)_{in} = D(\alpha_k, V_j)_{out}\,\forall\, V_j \in V$. Ensures that the computed outcome are valid circuits.

   (d) $\alpha'_k \subseteq \alpha_k$, i.e. the circuit α_k must consist of all edges in α'_k.

   (e) $\alpha'_k \cap \alpha_{k'} = \emptyset \mid \forall k' > k$, i.e. edges in α'_k should not be found in circuits of a higher f groups. If they are found in circuits of a higher f group circuit, they are already served at a higher frequency than required by its clustered group.

4. Solve the optimization problem [11] .

[10]With this, γ_k of higher k will consist of edges with higher f. To avoid confusion, note that the condition $k = min\,(k' \,,\, k_{\max})$ prevents having $k > k_{\max}$.

[11]Note that if k' is the number of non-empty groups, i.e. $k' = |k| \;\; \forall \gamma_k \neq \emptyset$, then the number of variables in this optimization is $k'|E|$. Meaning we only have to declare variables for α_k in actual implementation.

3.2.2 Combination Of Circuits

After having solved the optimization problem, it is now necessary to consider the combination of the circuits. The main reason for this is to prevent sub-optimum solution due to rounding up the number of agents needed [12] .

Recall that the optimization results in a number of disjoint circuits where connected circuits can be abstracted. Assuming that we have m number of connected circuits α''_m, let k_m indicate the group circuit α''_m belongs to, i.e. $\alpha''_m \subseteq \alpha_{k_m}$. The total circuit time τ'_m and the frequency at which the circuit α''_m should be visited by an agent are defined as:

$$\tau'_m = \sum_{\forall E_i \in \alpha''_m} \tau_{E_i}, \tag{3.10}$$

$$f_m = \max(f_{E_i}) | E_i \in \alpha''_m \cap \alpha'_{k_m}. \tag{3.11}$$

The number of agents n_m needed by α''_m and the number of agents $n_{m,m'}$ needed by the combination of α''_m and $\alpha''_{m'}$ are defined as:

$$n_m = \lceil f_m \tau'_m \rceil, \tag{3.12}$$

$$n_{m,m'} = \lceil \max(f_m, f_{m'})(\tau'_m + \tau'_{m'}) \rceil. \tag{3.13}$$

In addition, define $V(\alpha''_m)$ as the set of vertices incident to edges in α''_m. The algorithm for joining circuits is as follows:

```
   input: $\alpha''$, $\alpha'$, $f_{E_i}$
   output: $\alpha''$, $n$
1: for each $m$ do
2:     Initialize $f_m = \max(f_{E_i}) \forall E_i \in \{\alpha''_m \cap \alpha'_{k_m}\}$
3:     Initialize $\tau'_m = \sum_{\forall E_i \in \alpha''_m} \tau_{E_i}$
4: end for
5: while $n_{m,m'} \leq (n_m + n_{m'}) \exists m, m' \ni m' > m; V(\alpha''_m) \cap V(\alpha''_{m'}) \neq \emptyset$ do
6:     for each $m$ do
7:         Find $m'_m : \min_{m'_m}(n_{m,m'_m} - (n_m + n_{m'_m}))$
```

[12] For example, mathematically an edge needs 0.1 agent but since agents comes in integer numbers it is rounded up to 1 and therefore the edge is visited by an agent at a f 10 times higher than necessary.

```
          Subject to:
          $m'_m > m$
          $V(\alpha''_m) \cap V(\alpha''_{m'_m}) \neq \emptyset$
          $D(\alpha_k, V_j)_{in} = D(\alpha_k, V_j)_{out} \, \forall \, V_j \in V(\alpha''_m) \cup V(\alpha''_{m'_m})$
 8:     end for
 9:     if $\exists m, m'_m : n_m = n_{m'_m} = 1$ then
10:         Find $m : \min_{\forall m : n_m = n_{m'_m} = 1} \mid f_m - f_{m'_m} \mid$
11:     else
12:         Find $m : \min_{\forall m}(n_{m,m'_m} - (n_m + n_{m'_m}))$
13:     end if
14:     $n_m \leftarrow n_{m,m'_m}$;  $n_{m'_m} \leftarrow \emptyset$
15:     $f_m \leftarrow \max(f_m, f_{m'_m})$;  $f_{m'_m} \leftarrow \emptyset$
16:     $\alpha''_m \leftarrow \{\alpha''_m \cup \alpha''_{m'_m}\}$;  $\alpha''_{m'_m} \leftarrow \emptyset$
17:     $\tau'_m = \tau'_m + \tau'_{m'_m}$;  $\tau'_{m'_m} \leftarrow \emptyset$
18: end while
```

The algorithm takes in all α''_m and attempts to reduce the number of agents needed by combining circuits in pairs in iterations. This process continues until no improvement can be made. The index m'_m indicates the circuit for α''_m to join with for the highest amount of agent reduction. As we can see in line 9 to 13, emphasis is given to combining pair of circuits if both are assigned only one agent; if there was more than one of such pairs, emphasis is given to the pair of circuits that has the lowest different in their f_m value.

3.3 Approximate Algorithm For Linear Complexity Growth

While the solution presented in section 3.2 provides optimum solutions for a given $k_{\max}$ value, it is still high in terms of complexity (refer to 3.4). In order to overcome the exponential growth, an approximation algorithm is proposed.

The basic idea behind this approximation is to solve for each α_k, each as an individual optimization problem beginning from the highest k^{th} group to the lowest in iterations. In each iteration, we update $\alpha'_{k'} \, \forall \, 1 \leq k' \leq k$ with respect to the constraints stated in step 3 of section 3.2.1. In addition, the new objective function is[13]:

$$\sum_{\forall E_i \in \alpha_k} w_{E_i} \tau_{E_i}, \tag{3.14}$$

where the weight w_{E_i} is defined as:

$$w_{E_i} = \begin{cases} (f_k^{\max} - f_{k'}^{\max})^2 & \text{if } E_i \in \alpha'_{k'}; k \geq k' \\ f_k^{\max 2} & \text{otherwise} \end{cases}. \tag{3.15}$$

3.3.1 Combinatorial Optimization 2

Assuming same group assignment for γ_k as equation (3.7), the Approximate Algorithm is given as:

```
    input: $\gamma$, E, V
    output: $\alpha$
 1: for each $k$ do
 2:     Initialize $\alpha'_k = \gamma_k$
 3: end for
 4: for each $k$ from $k_{\max}$ to 1 do
 5:     for each $E_i \in E$ do
 6:         Calculate $w_{E_i}$
 7:     end for
 8:     Solve for $\alpha_k : \min_{\alpha_k}(\sum_{\forall E_i \in \alpha_k} w_{E_i} \tau_{E_i})$
        Subject to:
        $\alpha_k \subseteq E$
        $D(\alpha_k, V_j)_{in} = D(\alpha_k, V_j)_{out} \, \forall V_j \in V$
        $\alpha'_k \subseteq \alpha_k$
 9:     for each $k' : 1 \leq k' < k$ do
10:         $\alpha'_{k'} \leftarrow \alpha'_{k'} \setminus \alpha_k$
11:     end for
12: end for
```

[13]The weight w_{E_i} replaces the $f_k^{max\,2}$ term of the original objective function. The term $(f_k^{\max} - f_{k'}^{\max})^2$ explicitly indicates the tradeoff in using edges of lower frequency groups.

Note that the constraint $\alpha'_k \cap \alpha_{k'} = \emptyset \mid \forall k' > k$ of the original algorithm is missing. The missing condition is manually maintained in the loop in line 9 to 11, in iteration. Using this algorithm, we can find an approximate solution to the original algorithm. After which, the same *Combination of Circuits Algorithm* presented in section 3.2.2 can be applied.

3.4 Fixed Circuit Complexity

In the fixed circuit solution, we solve for the shortest disjoint circuits α'_k for each group k. The complexity upper bound is $2^{|E'|} : |E'| < |E|$. Typically $|E'| \ll |E|$ as we only consider edges in E near edges in α_k since we are solving for the shortest disjoint circuits, i.e. edges further away usually increase circuit lengths. However, it is still possible to artificially create a graph G where $|E'| \approx |E|$.

In the Original fixed circuit solution, all circuits are computed as a single optimization problem. The edge selection in the k^{th} group can affect the circuits in groups $k' < k$ since it affects $\alpha_{k'}$. Therefore the complexity is $O(2^{k|E'|})$. However in the Approximate Algorithm, these circuits are solved independently therefore the complexity is $O(k2^{|E'|})$.

3.5 Additional Agents In Approximate Algorithm

The Approximate Algorithm has a linear complexity growth with respect to k, but as a trade-off it may result in overlaps in edges in the circuits computed. This overlap comes from the approach of computing only the shortest disjoint circuit in iteration without considering circuit lengths of later iterations which then cause additional agents to be required. For example, when α_k is computed, some edges in $\alpha'_{k'} : k' < k$ should had been included to obtain the optimum solution. This cause the unused edges in $\alpha'_{k'}$ to require to form disjoint circuits in $\alpha_{k'}$ which has an overlap with α_k. However, as these edges in $\alpha'_{k'}$ is served at $f^{\max}_{k'} < f^{\max}_{k}$. Therefore, there is also some reduction in the amount of additional agents at the same time.

The actual amount of additional agents required depends on the structure of the graph G and can be quite unpredictable. However, there is an upper bound to this increase in agents. For instance, when $k_{\max} = 1$ there is no overlap since we solve for only one circuit. For all $k_{\max} > 1$, the upper bound of overlapped edges in terms of time it takes to traverse them is[14]:

$$\sum_{\forall E_i \in E} \frac{\tau_{E_i}}{k'}, \tag{3.16}$$

where $k' = |k|$: $\alpha_k \neq \emptyset$. Therefore the upper bound for number of additional agents needed is[15]:

$$\left\lceil \sum_{\forall k: \alpha_k \neq \emptyset} f_k^{\max} \sum_{\forall E_i \in E} \frac{\tau_{E_i}}{k'} \right\rceil, \tag{3.17}$$

where $k' = |k|$: $\alpha_k \neq \emptyset$.

Therefore, the additional agents in the Approximate Algorithm is 0 if $k_{\max} = 1$ and ranges from 0 to the upper bound stated in equation (3.17) for all $k_{\max} > 1$ [16].

3.6 Performance Guarantee

As both algorithms works on the same basic intuition, the described performance guarantee from this section is applicable to both algorithms. Define k_{opt} as the optimum number of groups to have. First we assume we have N optimum number of agents when $k_{\max} = k_{\text{opt}} = \infty$, we explore the performance of the algorithm as the number of categories of edges $k_{\max}$ varies. In addition, we make the conservative assumption that in each group γ_k there is only one edge with $f_{E_i} = f_k^{\max}$ and all other edges in γ_k have $f_{E_i} = f_k^{\min}$. Recall that $f^{\min}$ is the non-zero minimum. The highest frequency of k^{th} group can be approximated by:

$$\begin{array}{l} f_k^{\max} = f^{\min} + (k)\,\frac{f^{\max} - f^{\min}}{k_{\max}} \\ = \frac{(k) f^{\max} + (k_{\max} - k) f^{\min}}{k_{\max}} \end{array}, \tag{3.18}$$

[14]Overlap cannot be higher otherwise it contradict with the fact that α_k is a shortest disjoint circuit, i.e. if the overlap is $> \sum_{\forall E_i \in E} \frac{\tau_{E_i}}{k_{\max}}$, there exists edges which could be selected to form a shorter disjoint circuit.

[15]This upper bound is conservative since at least one circuit should not had been considered, i.e. a circuit cannot overlap over itself.

[16]Note that the upper bound in the number of agents needed for the Approximate Algorithm is capped at the worse-case upper bound stated in equation (3.20) of section 3.6.

while the lowest frequency of kth group can be approximated by:

$$\begin{aligned} f_k^{\min} &= f^{\min} + (k-1)\,\tfrac{f^{\max}-f^{\min}}{k_{\max}} \\ &= \tfrac{(k-1)f^{\max}+(k_{\max}-k+1)f^{\min}}{k_{\max}} \end{aligned} . \tag{3.19}$$

To find the worse-case upper bound, consider the number of agents needed for each group which is $f_k^{\max}/f_k^{\min}$ times higher when $k_{\max} \neq k_{\text{opt}}$. Assuming that all groups corresponding to $k > 1$ have only one agent and is a negligible compared to total number of edges. The worse-case upper bound is:

$$N\tfrac{f_1^{\max}}{f_1^{\min}} = N\tfrac{f^{\max}+(k_{\max}-1)f^{\min}}{(k_{\max})f^{\min}}. \tag{3.20}$$

Note that the worse-case upper bound when $k_{\max} = 1$ and $k_{\max} = \infty$ are:

$$N\tfrac{f^{\max}}{f^{\min}} \mid k_{\max} = 1, \tag{3.21}$$

$$\lim_{k_{\max}\to\infty} N\frac{f^{\max} + (k_{\max}-1)\,f^{\min}}{k_{\max}f^{\min}} = N. \tag{3.22}$$

In more general cases, tighter upper bound can be found. First we assume that total number of agents needed is even divided amongst all groups γ_k regardless of the value of $k_{\max}$ used. Then the number of agents needed for a given $k_{\max}$ excluding group γ_1 is:

$$\begin{aligned} &N/k_{\max}\sum_{\forall k;k\neq 1}^{k_{\max}} \frac{f_k^{\max}}{f_k^{\min}} \\ &= N/k_{\max}\sum_{\forall k;k\neq 1}^{k_{\max}} \frac{kf^{\max} + (k_{\max}-k)\,f^{\min}}{(k-1)\,f^{\max} + (k_{\max}-k+1)\,f^{\min}} \\ &= N\left(1 + \tfrac{1}{k_{\max}}\sum_{\forall k;k\neq 1}^{k_{\max}} \frac{1}{(k-1) + k_{\max}\frac{f^{\min}}{f^{\max}-f^{\min}}}\right) \\ &< N\left(1 + \tfrac{1}{k_{\max}}\left(\ln(k_{\max}) + 1.7\right)\right). \end{aligned} \tag{3.23}$$

Here, the term $\tfrac{1}{k_{\max}}\left(\ln(k_{\max}) + 1.7\right)$ refers to the amount of additional agents needed divided by N. We define it as a function:

$$N^{+}(k_{\max}) = \tfrac{1}{k_{\max}}(\ln(k_{\max}) + 1.7). \tag{3.24}$$

However, we know that $N^+(k_{\max})$ is zero when $k_{\max} = k_{\text{opt}}$, i.e. we should remove the assumption $k_{\text{opt}} = \infty$. Therefore a tighter upper bound is [17] :

$$N\left(1 + N^{+'}(k_{\max}) - N^{+'}(k_{\text{opt}})\right). \tag{3.25}$$

3.7 Fixed Circuit Simulation Results (NUS)

In this simulation, we make use of the scenario described in section 1.4.5 and shown in figure 1.7. Both the original fixed circuit algorithm described in section 3.2 and the approximate version described in section 3.3 is applied to the scenario. In the simulation, the value of $k_{\max}$, the total number of groups edges are clustered into, is varied from 1 to 8. When $k_{\max} = 8$ there are six non-empty group since there are only six unique f values. In figures 3.1, 3.2 and 3.3, we can see the simulation results in terms of number of agents and computational run time. As we can see in figure 3.1, the number of agents needed for both algorithms is the same. The upper bound plot corresponds to the general case in equation (3.25). The worse-case upper bound in equation (3.21) when $k_{\max} = 1$ is 126 which is far too loose considering we actually only need 28 agents.

In figure 3.2, we can see that the original algorithm has exponential growth in run time and it is at 600 seconds when $k_{\max} = 8$. In figure 3.3 we see a linear growth in run time and it is at 2.35 seconds when $k_{\max} = 8$. Both computations are completed on a system with 2.40 GHz.

In figure 3.4, we see four computed circuits and agent assignment when $k_{\max} = 6$. Each circuit includes both roads of either direction along each edge drawn in figure 3.4. It is interesting to note that the red solid line circuit needs 4 agents which is a significant number. Perhaps it means that we should simply install a sensor along AYE instead of sending agents on it. If we did that, the whole NUS campus only requires 10 cars (or even lesser if we also take out Clementi Road and West Coast Highway).

[17]Recall that the group γ_1 is excluded as it leads to a loose upper bound similar to the worse-case upper bound. The upper bound here is tighter but applies only to general cases.

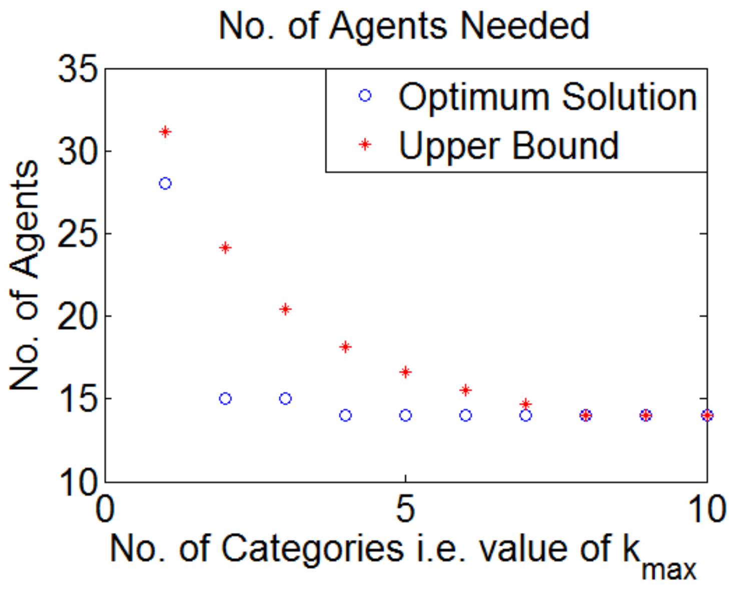


Figure 3.1: No. of Agents Needed Plotted Against Various Numbers of $k_{\max}$. Note that when $k_{\max} \geq k_{\text{opt}}$, no. of agents needed is always the same as the upper bound.

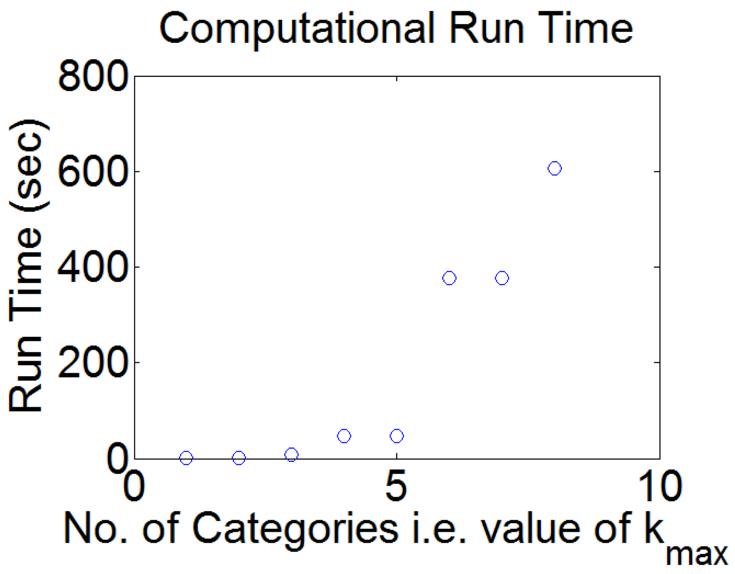


Figure 3.2: Computational Run Time for Original Algorithm Plotted Against Various Numbers of $k_{\max}$. On system: Dual-core 2.40 GHz, 3 GB RAM

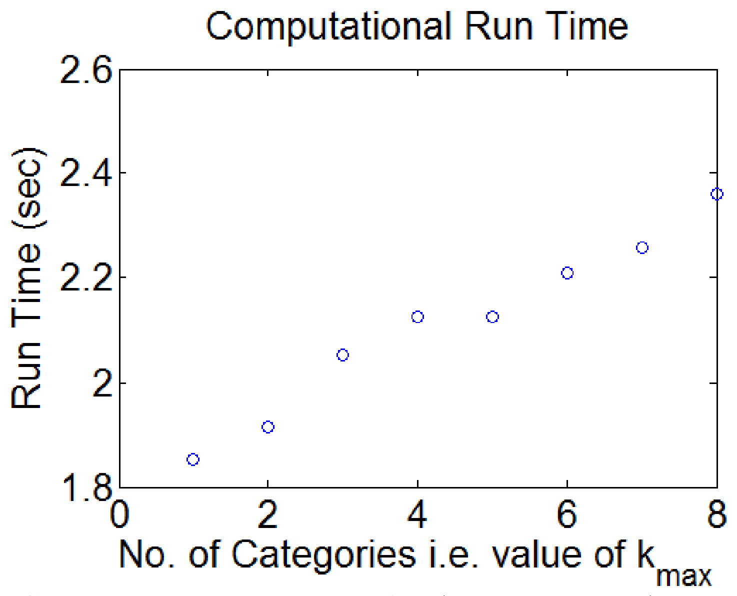


Figure 3.3: Computational Run Time for Approximated Algorithm Plotted Against Various Numbers of $k_{\max}$. On system: Dual-core 2.40 GHz, 3 GB RAM

3.8 Summary

In this chapter, we presented two versions of the fixed circuit solution, the first was the actual fixed circuit algorithm and the second was an approximate fixed circuit algorithm. We analyzed their performance and a simple NUS simulation was carried.

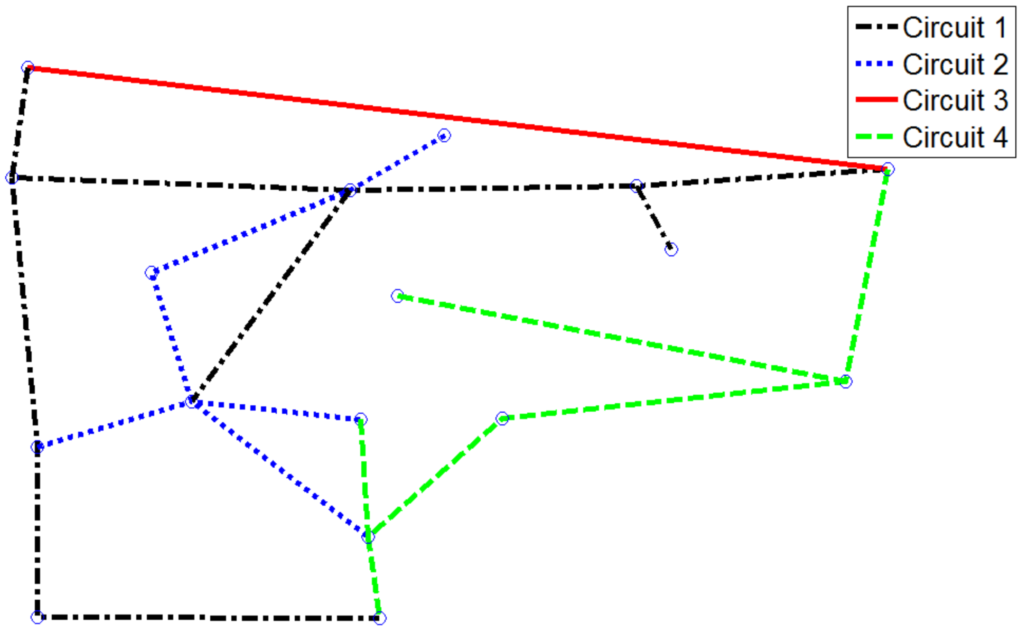


Figure 3.4: Four computed circuits to survey NUS when $k_{\max} = 6$ displayed in different. Agent assignment: Dashdot Line = 5; Dotted Line = 3; Solid Line = 4; Dashed Line = 2.

Chapter 4

Random Walk Solution

In this chapter, we look at the random walk Solution which offers some advantages over the fixed circuit solution despite the fact that it requires more agents and does not have a 100% guarantee that the objective $\tilde{g}(E_i, t) \leq \epsilon \; \forall E_i, t$ is always met. The first advantage is that random walk Solution does not restrict agents to a particular edge and therefore performance degrades more gracefully if any agent breaks down. This can be seen in section 4.3. The second advantage is that random walk Solution does not require explicit control in the agents' movement, i.e. agents do not have to ensure that they are evenly spread out. The third advantage is the flexibility to decide on the number of agents needed based on the confidence level at which the objective $\tilde{g}(E_i, t) \leq \epsilon \; \forall E_i, t$ is met.

4.1 Random Walk Without Communication Solution Algorithm

In this section, we explore a random walk Solution where we assume that all agents do not communicate with each other. This involves calculating N the total number of agents needed so that we satisfy the goal $\tilde{g}(E_i, t) \leq \epsilon \; \forall E_i, t$ at a confidence level a, assuming quasi-static problem as in 1.4.4. Define $n_{E_i} = \tau_{E_i} f_{E_i}$ as the minimum number of agents spread over edge E_i such that the goal is met. Let c_{E_i} be the road capacity of each edge E_i such that it represents the maximum number of cars that can be found on it. As each space on each edge E_i can only be occupied by one car at a time, we solve the hypergeometric distribution to calculate n'_{E_i}, the number of agents needed on each edge so that at a confidence level $\sqrt{a}$ there is n_{E_i} agents out of c_{E_i} spaces on the road spread out, i.e. with n'_{E_i} randomly placed agents there is a confidence level of $\sqrt{a}$ that n_{E_i} agents are spread out evenly on edge E_i.

We define probability function P where each p_{E_i} is given by:

$$p_{E_i} = n'_{E_i} / \sum_{E_i} n'_{E_i}. \tag{4.1}$$

In a Markov Chain, P corresponds to the equilibrium probability where each state E_i represents an edge and let T be the transition matrix. Each t_{E_i,E_j} in T represents the transitional probability for an agent on edge E_i to choose to move to edge E_j. Therefore

$$t_{E_i,E_j} \begin{cases} \geq 0 & \text{if } \exists V_i : D(E_i, V_i)_{in} = D(E_j, V_i)_{out} = 1 \\ = 0 & \text{otherwise} \end{cases}. \tag{4.2}$$

Markov Chain Monte Carlo method is commonly used to solve for the transition matrix T when given an equilibrium probability P. We can do it by applying any of the algorithms described in section 3 of [16].

However, it is necessary to ensure that the problem is feasible to begin with. There exists cases when given the equilibrium probability P finding a transition matrix T is not possible. For example, let P and T be given by

$$P = \begin{bmatrix} 0.9 \\ 0.05 \\ 0.05 \end{bmatrix} \quad T = \begin{bmatrix} 0 & t_{2,1} & t_{3,1} \\ t_{1,2} & 0 & t_{3,2} \\ t_{1,3} & t_{2,3} & 0 \end{bmatrix}.$$

Assuming that the directed edges in graph G have different entrance and exit nodes, then all values of the matrix T are 0. We note the relationship $TP = P$ and find the infeasible statement $0.05t_{2,1} + 0.05t_{3,1} = 0.9$. There is a contradiction here since probability is at most 1 and therefore the highest value for $t_{2,1}$ and $t_{3,1}$ is 1 thus $0.05t_{2,1} + 0.05t_{3,1} = 0.9$ is infeasible.

Let $S^{\mathrm{A}}(E_i)$ be a set of out-going edges incident to the entrance node for edge E_i inclusive of edge E_i. Let $S^{\mathrm{B}}(E_i)$ be a set of out-going edges incident to the exit node for edge E_i. A way to ensure that finding T is a feasible problem is to form a new matrix P' such that finding its transition matrix is possible. We first define all $n^{\mathrm{h}}_{E_j}$ values in the following way

$$n^{\mathrm{h}}_{E_j} \begin{cases} = \min(n'_{E_i}); \forall E_i \in S^{\mathrm{A}}(E_i) : n'_{E_i} \neq 0 & \text{if } n'_{E_j} = 0 \\ = n'_{E_j} & \text{otherwise} \end{cases}. \tag{4.3}$$

This definition ensures that no $n^{\mathrm{h}}_{E_j}$ value is equal to zero. With this, proceed to forming a matrix T' such that all out-going edges of each node has a probability proportional to its $n^{\mathrm{h}}_{E_j}$ out of the all out-going edges incident to its starting node. Each element of T' is

$$t'_{E_i,E_j}\begin{cases} = \frac{1}{|S^{\mathrm{B}}(E_i)|} & \text{if } E_j \in S^{\mathrm{B}}(E_i) \text{ and } \displaystyle\sum_{\forall E_k \in S^{\mathrm{B}}(E_i)} n^{\mathrm{h}}_{E_k} = 0 \\ = \dfrac{n^{\mathrm{h}}_{E_j}}{\displaystyle\sum_{\forall E_k \in S^{\mathrm{B}}(E_i)} n^{\mathrm{h}}_{E_k}} & \text{if } E_j \in S^{\mathrm{B}}(E_i) \text{ and } \displaystyle\sum_{\forall E_k \in S^{\mathrm{B}}(E_i)} n^{\mathrm{h}}_{E_k} \neq 0\,. \\ = 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Note that the equilibrium probability P is the normalized eigenvector (L2-Norm = 1) of transition matrix T with the eigenvalue corresponding to 1, therefore we can find P' using T'. With this we can modify all values of n'_{E_j} such that they produce P'. Making P' the new equilibrium probability, we know finding its transition matrix is a feasible problem since T' is already a solution to the problem. As we do not need to compute all eigenvectors we can use the method described here [19] for fast computation.

Using Chernoff's inequality we are able to calculate the total number of agents N so that at a confidence level $\sqrt{a}$ there are at least n'_{E_i} agents on each road. Recall that with n'_{E_i} randomly placed agents there is a confidence level of $\sqrt{a}$ that n_{E_i} agents are spread out evenly on edge E_i. Therefore, with N number of agents, there is a confidence level of a that the objective is met.

```
   input: $a$,$n_{E_i}$,$c_{E_i}$
   output: $P$,$N$
1: Find $\min(n'_{E_i})$:$\dfrac{\left\lfloor \frac{c_{E_i}}{\lceil n_{E_i} \rceil} \right\rfloor \binom{\lceil n_{E_i} \rceil}{\lceil n_{E_i} \rceil} \binom{c_{E_i} - \lceil n_{E_i} \rceil}{n'_{E_i} - \lceil n_{E_i} \rceil}}{\binom{c_{E_i}}{n'_{E_i}}} \geq \sqrt{a}$
2: $n'_{E_i} \leftarrow \frac{n'_{E_i} n_{E_i}}{\lceil n_{E_i} \rceil}$
3: Calculate $P : p_{E_i} = n'_{E_i} / \sum_{E_i} n'_{E_i}$
4: Calculate $n^{\mathrm{h}}_{E_i}$ based on equation (4.3)
5: Calculate $T' : t'_{E_i,E_j}$ based on equation (4.4)
6: Find $P'$ a normalized eigenvector (L2-Norm = 1) of $T'$ corresponding to
```

```
    eigenvalue of 1
7: $n'_{E_i} \leftarrow p'_{E_i} \max(n'_{E_j}/p'_{E_j});\ \forall E_j \in E$
8: Find $E'_i : p_{E'_i} = \max(p_{E_i}) \,\forall\, E_i$
9: Find $N$:$1 - e^{-\frac{(NP_{E'_i} - \lceil n'_{E'_i} \rceil)^2}{2NP_{E'_i}}} \geq \sqrt{a}; N > \left\lceil \frac{n'_{E'_i}}{p_{E'_i}} \right\rceil$
```

Note that n_{E_i} and n'_{E_i} may not be integers therefore it is necessary to use their ceiling in the calculations. There is an additional term:

$$\left\lfloor \frac{c_{E_i}}{\lceil n_{E_i} \rceil} \right\rfloor$$

apart from the hypergrometric distribution in line 1. This additional term represents the number of ways $\lceil n_{E_i} \rceil$ the number of agents can be evenly distributed along edge E_i. In line 2, n'_{E_i} is adjusted with respect to the real n_{E_i} value.

In line 7, we recalculate the n'_{E_i} values so that they produce the matrix P' if used to calculate the equilibrium probability. It is very important to note that the n'_{E_i} values in line 7 cannot be lower than that which is calculated in line 2 therefore in line 7 we have the term $\max(n'_{E_j}/p'_{E_j})$ to ensure that this constraint is not violated.

In lines 8 and 9, edge E'_i corresponding to the highest p_{E_i} is used. It is not necessary to consider all edges since all p_{E_i} are proportional to their respective n'_{E_i} values. The lower bound condition for N in line 9 is always true and is simply used to speed up search for N.

4.2 Random Walk With Communication Solution Algorithm

Here, we look at random walk with Communication Solution which is the same as the random walk Without Communication Solution except there are communication between agents and therefore we explicitly control the agents' movement. Meaning they are now assumed to spread evenly throughout each edge with respect to the amount of time it takes to traverse the edge. All other random walk advantages over fixed circuit solution is retained except this solution cannot be extended to the scenario where we install sensors on private cars and taxis.

All agents are assumed to have a communication range as far as the physical distance of any road; therefore all agents are able to communication with each other as long as they are on the same road. On each edge, agents are constantly informing each other of their respective location on the edge. Based on the amount of time it takes to traverse the edge τ_{E_i} and the total number of agents on the edge n''_{E_i}, they can speed up or slow down so that each agent takes τ_{E_i}/n''_{E_i} amount of time to reach the current location of its immediate neighbour that is in front of it along edge E_i.

The algorithm here is exactly identical to its counterpart the random walk Without Communication Algorithm in section 4.1, except line 1 is removed and line 2 is modified to $n'_{E_i} \leftarrow n_{E_i}$. It is given as follows:

```
   input: $a$,$n_{E_i}$,$c_{E_i}$
   output: $P$,$N$
1: $n'_{E_i} \leftarrow n_{E_i}$
2: Calculate $P : p_{E_i} = n'_{E_i} / \sum_{E_i} n'_{E_i}$
3: Calculate $n^{\mathrm{h}}_{E_i}$ based on equation (4.3)
4: Calculate $T' : t'_{E_i,E_j}$ based on equation (4.4)
5: Find $P'$ a normalized eigenvector (L2-Norm = 1) of $T'$ corresponding to
   eigenvalue of 1
6: $n'_{E_i} \leftarrow p'_{E_i} \max(n'_{E_j}/p'_{E_j});\ \forall E_j \in E$
7: Find $E'_i : p_{E'_i} = \max(p_{E_i}) \,\forall\, E_i$
8: Find $N$:$1 - e^{-\frac{(NP_{E'_i} - \lceil n'_{E'_i} \rceil)^2}{2NP_{E'_i}}} \geq \sqrt{a}; N > \left\lceil \frac{n'_{E'_i}}{p_{E'_i}} \right\rceil$
```

4.3 Random Walk Simulation Results (NUS)

A simulation was carried out based on the same environment in section 3.7 as shown in figure 1.7. In figure 4.2 we can see the capacity of each edge c_{E_i} is estimated as well but adjusted so that the results are as believable as possible. The values are decided based on both my personal experience and an assumption that vehicles follows traffic safety rule and keep a few car distances apart. For example, the section of AYE highway from Node 1 to Node 11 is about 2km long. Assuming cars are driving at 90km/hr which is the same as 25m/s. Assuming a driver typically has a reaction time of 1 second, the car would travel 25m before the driver steps on the brake. Assuming that cars are about 5m long, and assuming that it is possible for it to skid 40m before it is able to stop, this would give a safety distance of 70m. Therefore, it is assumed that a highway segment of 2km long can hold up to 29 cars; $2000/70 \approx 29$.

In figure 4.1, we can see the number of agents needed with respect to the confidence level we have in satisfying the goal $\tilde{g}(E_i, t) \leq \epsilon \ \forall E_i, t$. We can see that 295 agents are needed in order to have a confidence level of 95% when we have no communication while we only need 144 agents when there are communications between agents. We can also see that we always require lesser agents for the same level of confidence level when agents can communicate with each other. Computational time taken here is in the order of seconds. It is also interesting to note that if any of the agents were to break down, performance on the entire road network will degrade gracefully as the number of available agents varies with respect to the results of the plot in figure 4.1.

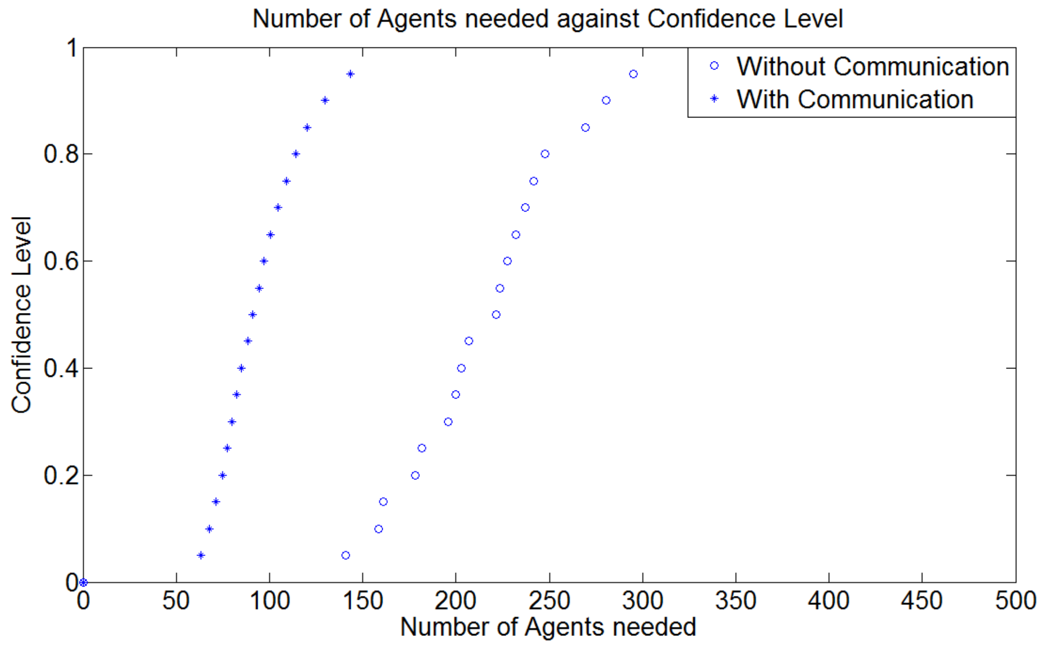


Figure 4.1: random walk Algorithm Siumation Results.

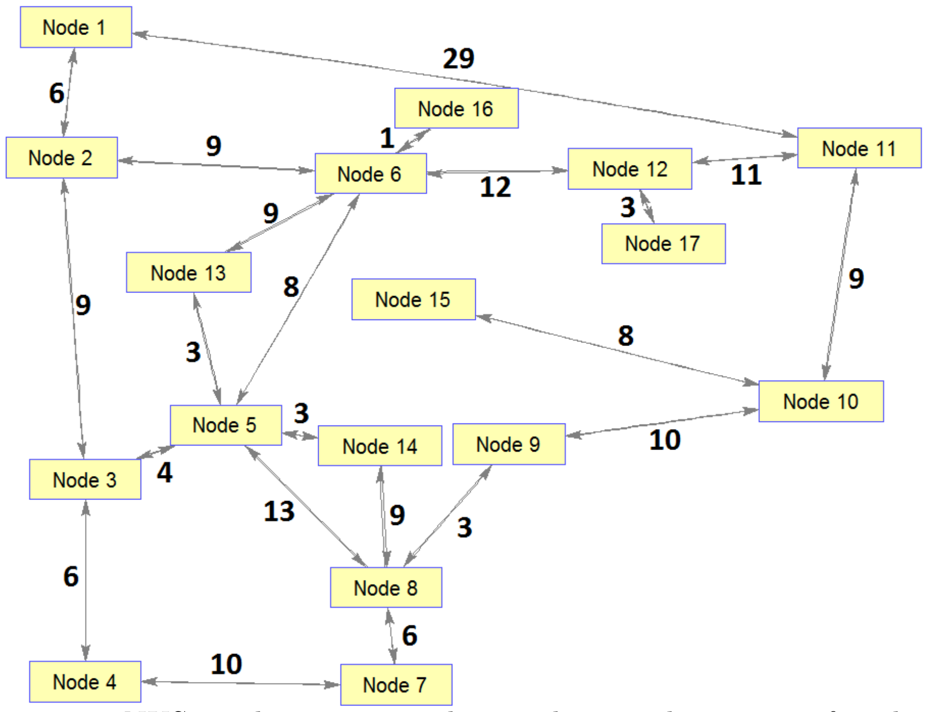


Figure 4.2: NUS road capacity. The number at the center of each edge corresponds to the number of cars the edge can hold.

4.4 Summary

In this chapter, two versions of the random walk algorithm were introduced. The first was meant for the scenario where agents were not allowed to communicate while the second was for the scenario where communication between agents along the same edge was allowed. It was found that the second version yields a less number of agents needed, however, the first version gave us an estimation of the number of privates cars in Singapore we can install sensors on to achieve the same effect.

Chapter 5

Singapore-Scale Simulations

In this chapter, we present the Singapore scale simulation carried out using the algorithms presented so far. The raw data are provided by LTA and ComfortDelgro, however, some processing were necessary so that useful information can be derived from them; more details can be found in section 5.1. The Singapore road network is shown in figure 5.1. In the simulations carried out, the original fixed circuit algorithm is excluded as its complexity is too high. However, the approximate fixed circuit algorithm is still used in the simulations and results are presented. In addition, the simulation was carried out on the quasi-static problem at 8am of weekdays as it is found to be the scenario where most number of agents is needed. This is sufficient for finding out the minimum number of agents required during any time and day to satisfy the problem objective stated in section 1.4.4.

5.1 Simulation Setup (Singapore)

In the Singapore scale simulation carried out, we use taxi data from a large fleet of taxis in Singapore provided by ComfortDelgro. It consists of four weeks of data (August 2010) from 16,000 taxis in Singapore, which amounts to approximately 31GB of data. Each taxi record contains the car id, the driver id, the time stamp, the GPS location, and status of taxi, i.e. available, person on board (hired), busy etc. Records are logged at intervals between 30 seconds and 2 minutes, depending on the network connectivity. In addition, details of the Singapore road network are provided by Land Transport Authority. Each road record includes the road name, road capacity, speed limit, GPS location of either end of the road's junction. There are a total of 77,283 roads and 42,323 road junctions in Singapore. The Singapore road network is shown in figure 5.1. The taxi GPS data are provided by ComfortDelgro and the road network data are provided by Land Transport Authority. These real data are used in the Singapore scale simulation.

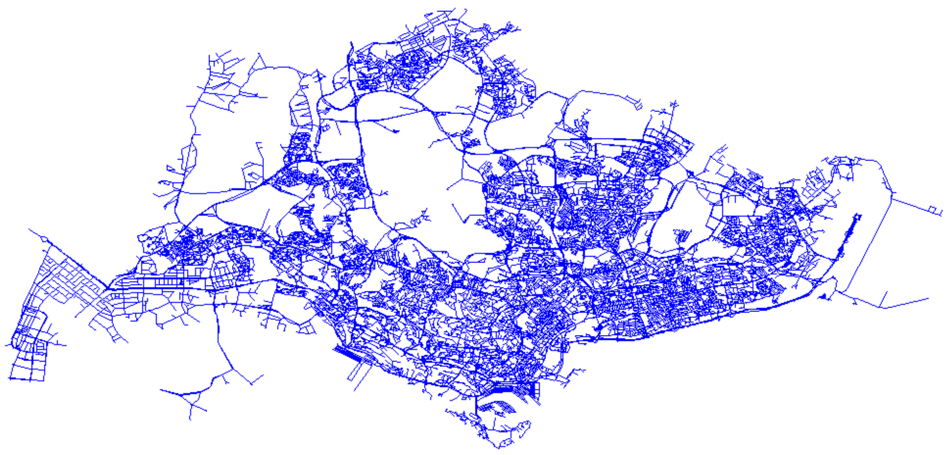


Figure 5.1: Singapore Road Network

5.1.1 Hidden Markov Map Matching

While the GPS information for each taxi is known, it is still necessary to interpret this information as road routes taken by each taxi. A naive approach is not feasible as GPS information can be noisy and intervals of 2 minutes can sometimes mean that the taxi have travelled over several roads. Therefore a Hidden Markov Model approach is adopted as in [13].

5.1.2 Differentiation Of Weekday And Weekend

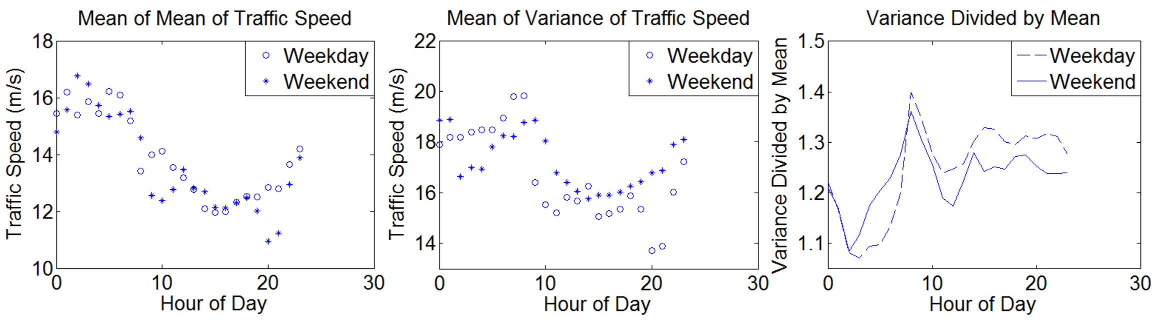
In effort to improve reliability of sampling of data, 9th of August is removed as travel pattern was expected to be different on a public holiday. In addition, the sampled data are classified into "weekday" and "weekend" groups. "Weekday" corresponds to sampling from the time 00:00 to 23:59

for each day from Monday to Friday while "weekend" corresponds to that of Saturday and Sunday from time 00:00 to 23:59.

5.1.3 Time Of Day For Highest Number Of Agents Required

Recall that the problem objective is $\tilde{g}(E_i, t) \leq \epsilon \ \forall E_i, t$. In order to identify the minimum number of agents needed to ensure that the objective is met it is sufficient to only consider the worst case scenario throughout the day. The highest number agents are required when the variances $\sigma^2_{t,E_i} \forall E_i, t$ are high and when $\mu_{t,E_i} \forall E_i, t$ are low since it mean that each edge needs to be visited more often while it takes longer to traverse these edges and therefore a high number of agents are required.

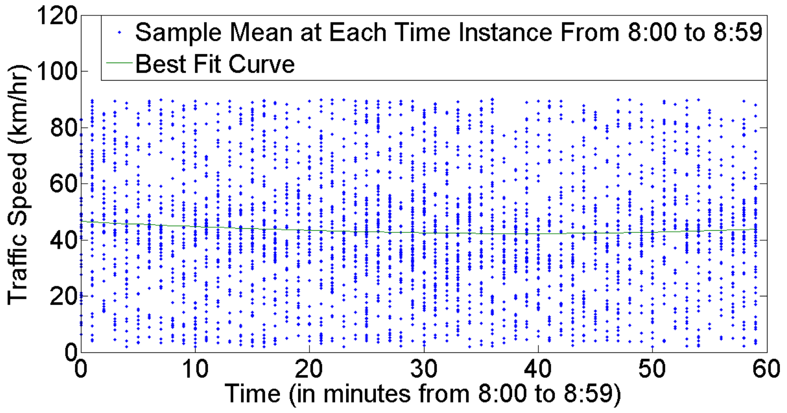
To identify the time of the day that requires the most number of agents, we first split the time of the day into 24 numbered sections starting from 0 to 23 and each of these section correspond to all times within that hour, i.e. section 0 corresponds to time 0:00 to 0:59 and therefore $t = \{0 \ldots 23\}$. We then find the mean of μ_{t,E_i} and σ^2_{t,E_i} overall E_i for each time t, let $\bar{\mu}_t$ and $\bar{\sigma}^2_t$ represent each of these respectively. We then compare the value of $\bar{\sigma}^2_t / \bar{\mu}_t$ and find the time of the day that corresponds to the highest value. The values of $\bar{\mu}_t$, $\bar{\sigma}^2_t$ and $\bar{\sigma}^2_t / \bar{\mu}_t$ are plotted in figure 5.2.



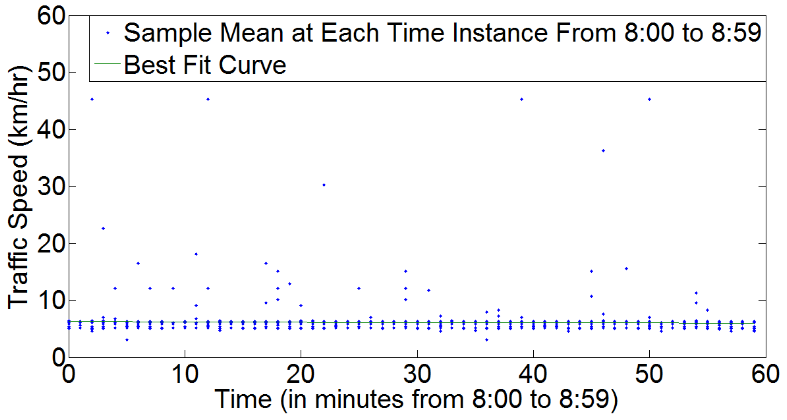
(a) Mean of mean of traffic speed (b) Mean of variance of traffic speed (c) Mean of variance divided by mean of mean

Figure 5.2: Plots of $\bar{\mu}_t$, $\bar{\sigma}^2_t$ and $\bar{\sigma}^2_t / \bar{\mu}_t$

From figure 5.2a, we can see that the congestion is worse during weekend from time 20:00 to 20:59. From figure 5.2b, the time of highest variance is during weekday from 08:00 to 09:59. From figure 5.2c, we can see that the time of highest need for agents is from 08:00 to 08:59 during the weekdays. Therefore, we only need to study the number of agents needed during weekdays 08:00 to 08:59 to understand the minimum number of agents needed to fulfil the problem objective.



(a) CTE Traffic Speed Sample Points (Total of 3,261 samples)



(b) ECP-Nicoll Highway Traffic Speed Sample Points (Total of 1,666 samples)

Figure 5.3: Traffic Sample Points

5.1.4 Growth Rate Of Entropy

In order to compute f_{E_i}, it is necessary to calculate the rate β_{E_i} from time 08:00 to 08:59. We define time $t = \{0 \dots 59\}$ of which each represents each minute of the hour from time 08:00 to 08:59, i.e. $t = 2$ correspond to time from 08:02:00 to 08:02:59. We then find the best fit for each edge E_i using all of the traffic speed sample points from 08:00:00 to 08:59:00. Let $f(E_i, t)$ represent the best fit curve for edge E_i and time $t = \{0 \dots 59\}$. For each t, we calculate the "variance" ${\sigma'}^2_{t,E_i}$. It differs from the original definition of variance as we take the value of $f(E_i, t)$ as the "mean" instead of using the actual mean at each time t. The value of β_{E_i} is then calculated as the mean of ${\sigma'}^2_{t,E_i}$ over all time t. This is interpreted as the amount of entropy growth per minute.

In figure 5.3, we can see some examples of the best fit curve and the distributions of samples. In figure 5.3a, there are a total of 3,261 samples and as we can see variance of traffic speed along CTE is very high. In figure 5.3b, we see that even though the road from ECP entering Nicoll Highway is usually very congested. We can see that the variance is actually very low with most of the 1,666 sample points compact together.

5.1.5 Entropy Threshold

After calculation of maximum entropy for all edges E_i using equation (1.3), it is found that the mean of all maximum entropy $\tilde{g}^{\max}(E_i) \forall E_i$ is 2.8 Nat. An entropy threshold $\epsilon = 1$ Nat was chosen for all edges in this thesis. Assuming that an edge has maximum speed $\hat{s}_{E_i} = 90$km/hr, minimum speed $\check{s}_{E_i} = 0$km/hr and mean speed $\mu_{t,E_i} = 60$km/hr. 1 Nat corresponds to $\sigma_{t,E_i} = 7.5$. This corresponds to a 50% probability that the speed lies between +/- 5km/hr around μ_{t,E_i} and a 82% probability that the speed lies between +/- 10km/hr. The traffic speed distribution of the edge described with uncertainty of 1 Nat is shown in figure 5.4.

5.1.6 Edge Frequencies

Using equation (1.6), we can calculate the frequencies at which each edge should be visited. For example, CTE needs to be visited once every 27 minutes, its traffic speed samples and best fit curve is shown in figure 5.3a. After calculation of frequencies we can calculate the time period at which each edge needs to be visited, it is found that 16,254 out of 77,283 edges does not need to be visited. Out of the remaining edges, 43,583 edges need to be visited at a period $<$ 1hour (this includes CTE that needs to be visited

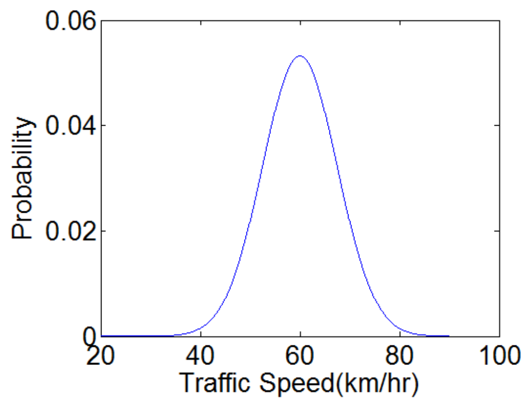


Figure 5.4: Traffic Speed Distribution for 1 Nat

once every 27 minutes); 16,703 edges need to be visited at a period between 1 to 3hours, 743 edges need to be visited at a period > 3hours.

5.2 Singapore-Scale Simulation: Relaxed Eulerian Circuit

In this simulation, the directed graph G represents the Singapore road network. The relaxed eulerian circuit Algorithm is applied so that pseudo-edges can be added, to satisfy the condition of $D^{+/-}(V_i) = 0 \forall V_i$. Below is a table that summarize the number of nodes, such that $D^{+/-}(V_i) \neq 0$.

$D^{+/-}(V_i) =$	-2	-1	1	2
$\lvert V_i \rvert$	32	3063	3073	27

Table 5.1: Number Of Unbalanced Nodes.

In table 5.1 we can see that the number of nodes is not symmetrical in the sense that the number of nodes with $D^{+/-}(V_i) = -2$ and the ones with $D^{+/-}(V_i) = 2$, which is not the same. This is the same for number of nodes with $D^{+/-}(V_i) = -1$ versus $D^{+/-}(V_i) = 1$. However, the total unbalanced degree still cancels out, i.e. $-2 \times 32 - 3063 = 3073 + 2 \times 27 = 3127$.

The computational complexity of the algorithm as we know is O(n^3), where n represents the number of unbalanced nodes multiplied by its degree 3127. However, after removal of source and sink nodes, (done by lines 1 to

4 of the algorithm in section 2.4.2) the number drops to 3075. The total time taken by the relaxed eulerian circuit Algorithm is 51 minutes and 54 seconds on a 2.4 GHz computer. The process where source and sink nodes are removed took 6 seconds. The amount of time taken to compute the shortest path between all unbalanced nodes to form a 3,075 by 3,075 cost matrix is 3 minutes and 52 seconds. The amount of time the node pairing process took is 47 minutes and 56 seconds.

5.3 Singapore-Scale Simulation : Approximate Fixed Circuit

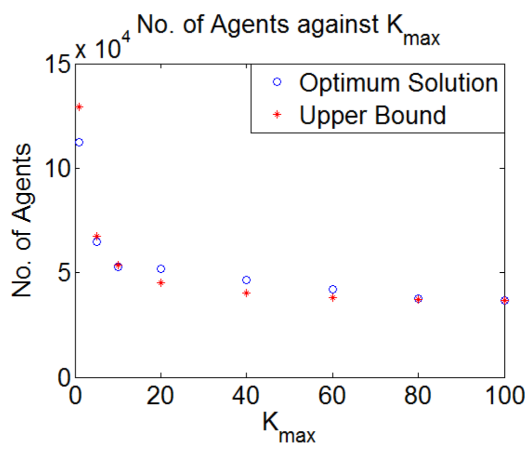
Recall that before the approximate fixed circuit algorithm is applied, it is assumed that the relaxed eulerian circuit Algorithm is applied. The simulations are carried out for a varying number of $k_{\max}$ values and the results of the simulation is shown in figure 5.5.

The optimum number of agents is found when $k_{\max} = 100$. The shape of graphs for both number of agents needed and computational time is similar to the ones found in section 3.7, where the simulation results for NUS is shown. The general case upper bound computed from equation (3.25) of section 3.6 is found to be lower than the actual solutions computed for some $k_{\max}$. However, it still provides a very close estimation of the actual number of agents required.

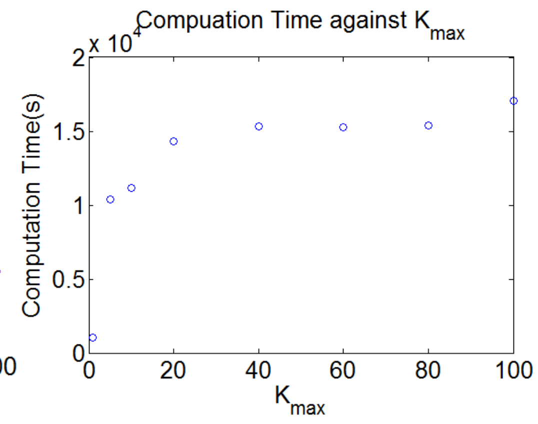
When $k_{\max} = 100$, the number of agents required is 36,669 while the amount of time it takes to compute it is 17,083 seconds (not including time taken for the relaxed eulerian circuit Algorithm) on a 2.4 GHz computer with 4GB Ram. This is about 4 hours and 45 minutes. Note that at least 4GB RAM is required to make the computation due to the high amount of memory required by in the optimization process of the approximate fixed circuit algorithm.

5.4 Singapore-Scale Simulation: Random Walk

A simulation was carried out using both versions of the random walk algorithm, with and without communication. The computation time is very short with an average of about 5 minutes and 7 seconds for the random walk without communication algorithm and 5 minutes and 2 seconds for the



(a) Number Of Agents Required



(b) Computation Time Taken

Figure 5.5: Approximate fixed circuit Simulation Results (Singapore)

random walk with communication algorithm for each confidence level on a 2.4 GHz computer, i.e. for each confidence level, we compute the number of agents needed and reported is the time taken for each of these computations. The results can be seen in figure 5.6.

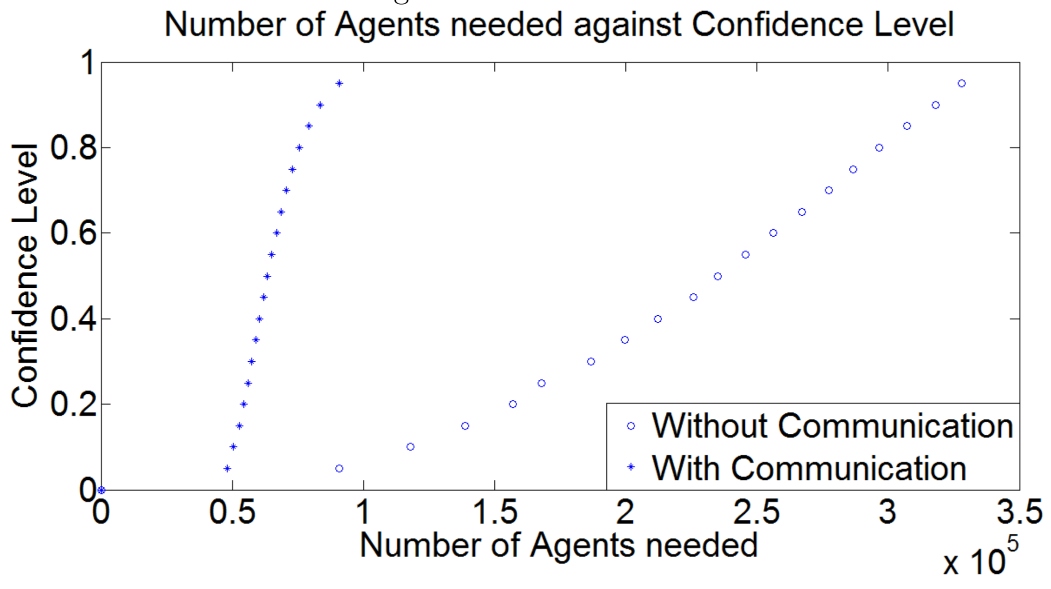


Figure 5.6: random walk Algorithm Siumation Results (Singapore).

Unlike the simulation carried out in section 4.3, the number of agents required without communication seems to increase relatively faster than in the NUS simulation. The difference indicates that in a large scale scenario, communication between agents serves to provide more advantage and the disadvantage due to lack of communication is more significant as well. As we can see in figure 5.6, 90,922 vehicles are required to achieve a confidence level of 95% when communication is allowed. When communication is not possible, 327,912 vehicles are required to achieve the same amount of confidence level.

We first assume that agent distribution is similar to that of the private cars in Singapore. This assumption while not always true is reasonable when uncertainty of traffic speed of each road corresponds to the number of cars that uses it. According to [11], there are about 570,207 private cars in Singapore. Therefore, if we were to install sensors on private cars so that we are able to make accurate traffic speed estimations for all roads in Singapore we need to do so for 327,912 out of 570,207 private cars in Singapore; which is approximately 57.5%.

5.5 Installation Of Sensors On Roads

An interesting point made in section 3.7 was that sensors can be installed on roads that require high number of agents or have a high frequency f_{E_i} to reduce the number of agents required. A short simulation was carried out to explore this idea. We first identify all edges that need to be visited by an agent once every 30 minutes or less based on the period calculated from their f_{E_i}. It was found that 12,538 out of 77,283 roads in Singapore fall under this category which is about 16.2%. We then set the frequencies of these roads to zero and compute the number of agents needed.

In the fixed circuit algorithm with $k_{\max} = 100$, it was found that only 1,151 agents are required. This is a huge decrease compared to the previously computed requirement of 36,669 agents when we do not consider installing sensors on roads. The required number of agents computed for the random walk without communication algorithm is 189,004 agents. Extending this result to the installation of sensor on private cars, it means that we now only need to install sensors on 189,004 private cars which is about 33.1%. The required number of agents computed for the random walk with communication algorithm is 28,792 agents [18]. There is no significant differ-

[18]Note that the number of agents required by random walk simulations assumes that a confidence level of 95% is required.

ence in computation time required in the three simulations. Out of all three simulations, the one that has the most significant improvement made is the fixed circuit algorithm.

5.6 Summary

In this chapter, we discussed about the data provided by LTA and ComfortDelgro and how they were processed into useful relevant information for the probe allocation problem simulation. Simulations had been carried out for the relaxed eulerian circuit Algorithm, the approximate fixed circuit algorithm and both versions of the random walk algorithm. In addition, we had also considered the case where sensors were installed on roads that require agents to visit them at a high frequency and a simulation was carried out for this scenario.

Chapter 6

Bus Route Planning Problem

The main similarity between the bus route planning problem and the probe allocation problem is the fact that they can be solved by planning a number of fixed circuit for buses/agents to traverse repeatedly in. The constant increase in the number of commuters at a bus stop is analogous to the constantly increasing uncertainty in traffic estimation. The reduction of the number of commuters at a bus stop is also similar to how uncertainty is decreased when an agent visits an edge. Due to the many similarities, it is believed that the fixed circuit algorithm can be modified to solve for the bus route planning problem.

6.1 Bus Route Problem Statement

Let V^{B} be a set of vertices with each vertex $v_i^{\mathrm{B}} \in V^{\mathrm{B}}$ corresponding to a bus stop. Let P^{B} be the demand matrix such that each $p_{i,j}^{\mathrm{B}}$ represents the number of commuters travelling from bus stop v_i^{B} to ${}^{\mathrm{B}}v_j$ [19]. Let $G^{\mathrm{r}} = (V^{\mathrm{B}}, E^{\mathrm{r}})$ be the graph representing a map such that each $v_i^{\mathrm{B}} \in V^{\mathrm{B}}$ is a bus stop while each $e_{i,j}^{\mathrm{r}} \in E^{\mathrm{r}}$ is the road between bus stop v_i^{B} and ${}^{\mathrm{B}}v_j$. Let $V^{\mathrm{B}\prime} \subseteq V^{\mathrm{B}}$ be the set of bus stops that are also bus terminals. In addition, let $G^{\mathrm{w}} = (V^{\mathrm{B}}, E^{\mathrm{w}})$ be another map such that each $e_{i,j}^{\mathrm{w}} \in E^{\mathrm{w}}$ is the walking path between bus stop v_i^{B} and v_j^{B}. The amount of time it takes to traverse edges in E^{r} and E^{w} is $c_{i,j}^{\mathrm{r}}$ and $c_{i,j}^{\mathrm{w}}$ respectively. Lastly, let $G^{\mathrm{B}} = (V^{\mathrm{B}}, E^{\mathrm{B}})$ be

[19]Can also be seen as the rate of increase in number of commuters travelling from bus stop i to j. This does not make a difference to the solution proposed.

the combination of G^{r} and G^{w} such that

$$e^{\mathrm{B}}_{i,j} = \begin{cases} e^{\mathrm{r}}_{i,j} & \text{if } c^{\mathrm{r}}_{i,j} \leq c^{\mathrm{w}}_{i,j} \\ e^{\mathrm{w}}_{i,j} & \text{if } c^{\mathrm{r}}_{i,j} > c^{\mathrm{w}}_{i,j} \\ \emptyset & \text{if } c^{\mathrm{r}}_{i,j} = c^{\mathrm{w}}_{i,j} = \infty \end{cases} .$$

Let α^{B} be a set of bus route such that each $\alpha^{\mathrm{B}}_m \in \alpha^{\mathrm{B}}$ consists of edges in E^{r}. Let $\tau^{\mathrm{B}\prime}_{i,j}$, $\tau^{\mathrm{B}\prime\prime}_{i,j}$ and $w_{i,j}$ be the total in-vehicle travel time, total walking time, and total bus waiting time for each commuter travelling from bus stop v^{B}_i to v^{B}_j respectively. Let $c(G^{\mathrm{B}})$ be a function that calculates the number of components in graph G^{B} including vertices with no edges incident to it while $c^{\min}(G^{\mathrm{B}})$ is a function that returns the number of component in graph G^{B} excluding vertices that are not incident to any edge. Let $d^{V^{\mathrm{B}\prime}}(\alpha^{\mathrm{B}}_m)$ be a function that returns the maximum degree of all vertices $v^{\mathrm{B}}_i \in V^{\mathrm{B}\prime}$ in the subgraph induced by edges in α^{B}_m, i.e. the maximum degree out of all nodes that are bus terminals in the bus route α^{B}_m.

The problem is to design a set of bus routes α^{B} such that the total travel time for all commuters are minimized, i.e.

$$\begin{array}{l} \min \left(\sum\limits_{\forall i,j} p^{\mathrm{B}}_{i,j} \left(\tau^{\mathrm{B}\prime}_{i,j} + \tau^{\mathrm{B}\prime\prime}_{i,j} + w_{i,j} \right) \right) \\ \text{Subject to:} \\ c(G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, \{\alpha^{\mathrm{B}}\})) = 1 \\ c^{\min}(G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, \alpha^{\mathrm{B}}_m)) = 1; \forall \alpha^{\mathrm{B}}_m \in \alpha^{\mathrm{B}} \\ D(\alpha^{\mathrm{B}}_m, v^{\mathrm{B}}_i)_{\mathrm{in}} = D(\alpha^{\mathrm{B}}_m, v^{\mathrm{B}}_i)_{\mathrm{out}}; \forall \alpha^{\mathrm{B}}_m \in \alpha^{\mathrm{B}}; v^{\mathrm{B}}_i \in V^{\mathrm{B}} \\ d^{V^{\mathrm{B}\prime}}(\alpha^{\mathrm{B}}_m) > 0; \forall \alpha^{\mathrm{B}}_m \in \alpha^{\mathrm{B}} \end{array} .$$

The first constraint ensures that the edges in all bus routes induce a graph with a single component inclusive of all nodes with no edges incident to them. This indicates that all bus stops are served by at last one bus route and can reach any other bus stops. The second constraint indicates that all bus routes induce a single component excluding nodes with no edges incident to them. The third constraint indicates that the in-degree and out-degree of all nodes in all bus routes are balanced. The second and third constraint together implies that all bus routes are circuits that form a single component and therefore can be implemented in real life, i.e. if there were more than one component it means the bus route consists of two disjoint circuits and if the in-degree and out-degree is not balanced it implies that the bus route does not form a circuit. The fourth constraint indicates that all bus routes visits at least one bus terminal.

6.2 Bus Route Algorithm

Recall that the probe allocation algorithm forms clusters of edges with similar frequencies, forms circuits that consists of edges in the same cluster, and then combines these circuits to reduce the number of agents needed. However, the bus route problem is slightly different in nature and direct application of the probe allocation algorithm is not possible.

The bus route algorithm consists of three main parts. The first part calculates the shortest circuit consisting of the shortest path between each origin-destination pair (OD pair). These shortest circuits are analogous to each edge of the probe allocation problem and form the basic circuits with which we form the bus routes. These circuits have to be the shortest possible since it is it helps ensure that bus routes would be short as well. However, these circuits may consist of walking paths and therefore requires modification. In addition, we only do so for OD pair with significant travel demand as we only wish to identify the most significant travel trends out of all OD pairs, i.e. $p^{\mathrm{B}}_{i,j}$ larger than mean of all non-zero values in P^{B}. After which, we consider a subset of the remaining OD pairs so that all bus stops are considered as either a source or a destination at least once. The remaining OD pairs are not considered as they are seen as noise in the demand matrix P^{B}. Note that the circuits computed may consist of edges corresponding to walking paths.

The second part of the algorithm consists of a few processes. Firstly, the walking paths of each circuit computed in the first part of the algorithm are removed. We then compute the shortest circuit that not only consists of the remaining edges after removal of walking paths but also only consists of edges corresponding to roads, i.e. we modify the circuits computed from the first part of the algorithm so that they do not consist of walking paths. The reason to do so is to form the basic circuit with which bus routes can be made from, therefore, walking paths are not allowed. Secondly, we cluster/combine similar circuits. The similarities of circuits are decided by the demand for each circuit, i.e. the expected number of commuters each circuit serve. This is analogous to the fixed circuit algorithm. Thirdly, we modify the circuits so that all of them pass by at least one bus terminal. Since buses are parked in bus terminals, it ensures that their circuits have proper start/end points. Fourthly, we remove bus routes that are considered to be insignificant in demand with the constraint that all bus stops are served by at least one of the remaining bus routes. The reason for this is to remove bus routes that are not required by a significant number of commuters as they are seen as noise in the set of bus routes. Similar to the reason why not every OD pairs are considered in the first part of the algorithm. Doing so serves to concentrate resources on bus routes that have high demand.

After the second part of the algorithm, the bus routes are already computed and finalized. The third part of the algorithm determines the number of buses on each bus route by estimating the actual number of commuters expected to be served on each bus route.

6.3 Algorithm Part 1

Let k be a number of selected OD pair, let each o_k and d_k represent the respective origin and destination bus stops and let $\alpha_k^{\mathrm{B}\prime}$ be the set of edges needed to be traversed such that it is the shortest circuit consisting the shortest path from o_k to d_k in G^{B}. Let $\gamma^v(G^{\mathrm{B}}, i, j)$ and $\gamma^e(G^{\mathrm{B}}, i, j)$ be the set of vertices and edges arranged in sequence with respect to the shortest path from v_i^{B} to v_j^{B} in graph G^{B} respectively.

Let k be a number of selected OD pair such that the selected OD pair satisfies two condition. Firstly, it includes all OD pair that has a significant amount of commuters travelling from the respective origin and destination. It is defined as OD pair that has equal or more commuter than the mean of all non-zero elements in P^{B}, i.e. $p_{i,j}^{\mathrm{B}}$ is selected if $p_{i,j}^{\mathrm{B}} \geq \bar{p}$ such that

$$\bar{p} = \frac{\sum p_{i,j}^{\mathrm{B}}}{|p_{i,j}^{\mathrm{B}}|}; \forall p_{i,j}^{\mathrm{B}} \neq 0.$$

Secondly, we consider some of the remaining OD pairs so that all bus stops are considered as either a source or a destination at least once. The second condition can be interpret as $c(G^{\mathrm{B}\prime}) = 1$ where $G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, \{\alpha^{\mathrm{B}\prime}\})$. As such, we will be able to consider OD pairs such that all bus stops are serviced by at least one bus route and that the most significant travel trends are considered while the insignificant ones are ignored.

The first part of the algorithm is as follows.

```
    input: $P^{\mathrm{B}}$, $G^{\mathrm{B}}$
    output: $\alpha^{\mathrm{B}\prime}$, $o$, $d$
 1: Sort all OD pair with respect to $p^{\mathrm{B}}_{i,j}$
 2: $k \leftarrow 1$
 3: Set $i$ and $j$ such that $p^{\mathrm{B}}_{i,j}$ correspond to the largest $p^{\mathrm{B}}_{i,j}$ value.
 4: while $p^{\mathrm{B}}_{i,j} \geq \bar{p}$ do
 5:     $o_k \leftarrow i$
 6:     $d_k \leftarrow j$
 7:     Find $\gamma^e(G^{\mathrm{B}}, i, j)$ using Dijkstra's algorithm : $\min\left(\sum_{e^{\mathrm{B}}_{i,j} \in \gamma^e(G^{\mathrm{B}},i,j)} c^{\mathrm{B}}_{i,j}\right)$
        Subject to:
        $D^{+/-}(d_k) = 1$
        $D^{+/-}(o_k) = -1$
        $D^{+/-}(v^{\mathrm{B}}_i) = 0 \ \forall v^{\mathrm{B}}_i \in V^{\mathrm{B}} \setminus \{o_k, d_k\}$
 8:     Find $\alpha^{\mathrm{B}\prime}_k$ : $\min\left(\sum_{e^{\mathrm{B}}_{i,j} \in \alpha^{\mathrm{B}\prime}_k} c^{\mathrm{B}}_{i,j}\right)$
        Subject to:
        $\gamma^e(G^{\mathrm{B}}, i, j) \subseteq \alpha^{\mathrm{B}\prime}_k$
        $D^{+/-}(v^{\mathrm{B}}_i) = 0 \ \forall v^{\mathrm{B}}_i \in V^{\mathrm{B}}$
 9:     $k \leftarrow k + 1$
10:     Set $i$ and $j$ such that $p^{\mathrm{B}}_{i,j}$ correspond to the next largest $p^{\mathrm{B}}_{i,j}$ value.
11: end while
12: while $\exists v^{\mathrm{B}}_i \notin o \bigcup d$ do
13:     Find $i, j : \max_{i,j}(p^{\mathrm{B}}_{i,j})$
        Subject to:
        $(v^{\mathrm{B}}_i \notin o \bigcup d) \vee (v^{\mathrm{B}}_j \notin o \bigcup d)$
14:     $o_k \leftarrow i$
15:     $d_k \leftarrow j$
16:     Find $\gamma^e(G^{\mathrm{B}}, i, j)$ using Dijkstra's algorithm : $\min\left(\sum_{e^{\mathrm{B}}_{i,j} \in \gamma^e(G^{\mathrm{B}},i,j)} c^{\mathrm{B}}_{i,j}\right)$
        Subject to:
        $D^{+/-}(d_k) = 1$
        $D^{+/-}(o_k) = -1$
        $D^{+/-}(v^{\mathrm{B}}_i) = 0 \ \forall v^{\mathrm{B}}_i \in V^{\mathrm{B}} \setminus \{o_k, d_k\}$
```

```
17:     Find $\alpha_k^{B\prime} : \min \left( \sum_{e_{i,j}^{B} \in \alpha_k^{B\prime}} c_{i,j}^{B} \right)$
        Subject to:
        $\gamma^e(G^B, i, j) \subseteq \alpha_k^{B\prime}$
        $D^{+/-}(v_i^B) = 0 \ \forall v_i^B \in V^B$
18:     $k \leftarrow k + 1$
19: end while
```

Recall that each o_k and d_k represents the respective origin and destination bus stops and $\alpha_k^{B\prime}$ is the set of edges needed to be traversed such that it is the shortest circuit consisting the shortest path from o_k to d_k in G^B. The variable k is simply used to append these set therefore its value is initialized as 1 in line 2 and increases by one in lines 9 and 18. In lines 5 to 8 and 14 to 17, we append the considered OD pair's o_k, d_k and $\alpha_k^{B\prime}$. In line 1, we sort the OD pairs so that they can be simply evaluated in the while loop from lines 3 to 8. The while loop in lines 4 to 11 considers the OD pairs with significant demand and the while loop in lines 12 to 19 considers some of the remaining OD pair such that $c(G^{B\prime}) = 1$ where $G^{B\prime} = (V^B, \{\alpha^{B\prime}\})$.

6.4 Algorithm Part 2

Let each $\alpha_m^B \in \alpha^B$ be a circuit consisting only of edges in E^r. Let each $P_m^{B\prime}$ of $P^{B\prime}$ be a matrix such that each element $p_{m,i,j}^{B\prime}$ is the total number of commuters expected to traverse the edge $e_{i,j}^r$ of circuit α_m^B. Let $d(\alpha_m^B)$ be the function that returns the maximum degree of all vertices $v_i^B \in V^B$ in the subgraph induced by edges in α_m^B. Define the function $\sigma(P^B)$ such that it returns the variance of the set of all non-zero elements in matrix P^B. Let $\max(P^B)$ be a function that returns the largest element in the matrix P^B. Let m' be a set of indices that refers to sets in α^B. Let $e(G, q)$ be the set of edges in the qth component in graph G.

The second part of the algorithm is as follows.

```
    input: $\alpha^{B\prime}$, $o_k$, $d_k$, $E^r$, $E^w$, $P^B$
    output: $\alpha^B$
1: $m \leftarrow 1$
2: for each $\alpha_k^{B\prime}$ do
```

```
3:     for each component $q$ in $G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, (\alpha_k^{\mathrm{B}\prime} \setminus E^{\mathrm{w}}))$ do
4:         Find $\alpha_m^{\mathrm{B}} : \min\left(\sum_{e_{i,j}^{\mathrm{r}} \in \alpha_m^{\mathrm{B}}} c_{i,j}^{\mathrm{r}}\right)$
           Subject to:
           $e(G^{\mathrm{B}\prime}, q) \subseteq \alpha_m^{\mathrm{B}}$
           $D^{+/-}(v_i^{\mathrm{B}}) = 0 \ \forall v_i^{\mathrm{B}} \in V^{\mathrm{B}}$
5:         $p_{m,i,j}^{\mathrm{B}\prime} \leftarrow p_{o_k,d_k}^{\mathrm{B}}; \forall e_{i,j}^{\mathrm{r}} \in \alpha_m^{\mathrm{B}}$
6:         $m \leftarrow m + 1$
7:     end for
8: end for
9: while $\exists \alpha_m^{\mathrm{B}}, \alpha_{m'}^{\mathrm{B}} : m \neq m'$ that can be combined such that $d(\{\alpha_m^{\mathrm{B}}, \alpha_{m'_m}^{\mathrm{B}}\}) <$
   $6$ do
10:    for each $\alpha_m^{\mathrm{B}}$ do
11:        Find $m'_m : \min_{m'_m}(\sigma(P_m^{\mathrm{B}\prime} + P_{m'_m}^{\mathrm{B}\prime}))$
           Subject to:
           $m'_m \neq m$
           $d(\{\alpha_m^{\mathrm{B}}, \alpha_{m'_m}^{\mathrm{B}}\}) < 6$
           $c^{\min}(G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, \{\alpha_m^{\mathrm{B}}, \alpha_{m'_m}^{\mathrm{B}}\})) = 1$
           $D(\{\alpha_m^{\mathrm{B}}, \alpha_{m'_m}^{\mathrm{B}}\}, v_i^{\mathrm{B}})_{\mathrm{in}} = D(\{\alpha_m^{\mathrm{B}}, \alpha_{m'_m}^{\mathrm{B}}\}, v_i^{\mathrm{B}})_{\mathrm{out}}; \forall v_i^{\mathrm{B}} \in V^{\mathrm{B}}$
12:    end for
13:    Sort $m'$ with respect to $\sigma(P_m^{\mathrm{B}\prime} + P_{m'_m}^{\mathrm{B}\prime})$
14:    Set $m'_m$ such that $\sigma(P_m^{\mathrm{B}\prime} + P_{m'_m}^{\mathrm{B}\prime})$ correspond to the lowest value.
15:    while $m' \neq \emptyset$ do
16:        $\alpha_m^{\mathrm{B}} \leftarrow \{\alpha_m^{\mathrm{B}}, \alpha_{m'_m}^{\mathrm{B}}\}$
17:        $P_m^{\mathrm{B}\prime} \leftarrow P_m^{\mathrm{B}\prime} + P_{m'_m}^{\mathrm{B}\prime}$
18:        $\alpha_{m'_m}^{\mathrm{B}} \leftarrow \emptyset$
19:        $P_{m'_m}^{\mathrm{B}\prime} \leftarrow \emptyset$
20:        $m' \leftarrow m' \setminus \{m, m'_m\}$
21:        Set $m'_m$ such that $\sigma(P_m^{\mathrm{B}\prime} + P_{m'_m}^{\mathrm{B}\prime})$ correspond to the next lowest
   value.
22:    end while
23: end while
24: for each $m$ do
25:    Find $\alpha_m^{\mathrm{B}} : \min\left(\sum_{e_{i,j}^{\mathrm{r}} \in \alpha_m^{\mathrm{B}}} c_{i,j}^{\mathrm{r}}\right)$
```

```
        Subject to:
        $d^{V^{\mathrm{B}\prime}}(\alpha^{\mathrm{B}}_{m}) > 0$
        $d(\alpha^{\mathrm{B}}_{m}) < 6$
        Original $\alpha^{\mathrm{B}}_{m} \subseteq$ Updated $\alpha^{\mathrm{B}}_{m}$
        $D^{+/-}(v^{\mathrm{B}}_{i}) = 0\ \forall v^{\mathrm{B}}_{i} \in V^{\mathrm{B}}$
        $c^{\min}(G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, \{\alpha^{\mathrm{B}}_{m}, \alpha^{\mathrm{B}}_{m'_m}\})) = 1$
        $D(\{\alpha^{\mathrm{B}}_{m}, \alpha^{\mathrm{B}}_{m'_m}\}, v^{\mathrm{B}}_{i})_{\mathrm{in}} = D(\{\alpha^{\mathrm{B}}_{m}, \alpha^{\mathrm{B}}_{m'_m}\}, v^{\mathrm{B}}_{i})_{\mathrm{out}}; \forall v^{\mathrm{B}}_{i} \in V^{\mathrm{B}}$
26: end for
27: Sort $\alpha^{\mathrm{B}}_{m}, P^{\mathrm{B}\prime}_{m}$ with respect to $\max(P^{\mathrm{B}\prime}_{m})$ such that $\max(P^{\mathrm{B}\prime}_{1})$ is the largest
28: Find minimum set $m'$
    Subject to:
    $m \subseteq m' \mid \forall \max(P^{\mathrm{B}\prime}_{m}) \geq \frac{1}{2}\max(P^{\mathrm{B}\prime}_{1})$
    $c(G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, \{\alpha^{\mathrm{B}}_{m'}\})) = 1$
29: Update $\alpha^{\mathrm{B}} \leftarrow \alpha^{\mathrm{B}}_{m'}$
```

The last two constraints lines 11 and 25 ensure that each bus route α^{B}_{m} induces a single component excluding nodes with no edges incident to them and that the in-degree and out-degree of all nodes are balanced. Together, the constraints ensure that each bus route is a single circuit that buses can traverse on. For example, if there were more than one component it means the bus route consists of two disjoint circuits and if the in-degree and out-degree is not balanced it implies that the bus route does not form a circuit.

Recall that there are 4 processes in the second part of the algorithm. Firstly, we modify the circuits computed from the first part of the algorithm so that they do not consist of walking paths. The circuits created after this modify are analogous to the circuits in the fixed circuit algorithm and they will be used to create bus routes after they have been clustered/combined. The circuit modification process runs from lines 1 to 8. As removing walking path may induce multiple components we handle each of the components in the loop stated in line 3. The variable m here serves as an index for each modified circuit computed in line 4. In line 5, we collect the demand information for each modified circuit.

The second process is where we cluster/combine similar circuits and it goes from lines 9 to 23. This is similar to the combination of circuits in the fixed circuit algorithm. One constraint for clustering/combining circuits is $d(\{\alpha^{\mathrm{B}}_{m}, \alpha^{\mathrm{B}}_{m'_m}\}) < 6$. This means that all bus stops are served at most 2 times (served once for each in-coming and out-going edge pair) from either direction by each bus route. The purpose of this constraint is to reduce the complexity of the bus routes and ensure its directness. Directness of bus route is a considered factor in some papers [24–26]. In lines 10 to 12, for each modified circuit, we find another modified circuit that it can be combined with such that the pair corresponds to the minimum amount of variance in the resulting combined demand matrix. This would imply that the circuits are similar analogous to the clustering/combining criteria in the fixed circuit algorithm. In lines 13 to 22, we do the actual combination of modified circuits. Here we adopt a greedy approach such that we begin with the pair of modified circuits m and m'_m that would combine to result in a demand matrix that corresponds to the minimum variance out of all combinations. Modified circuits are allowed to be combined only once in each iteration to ensure that all combinations of circuits comply to the constraint $d(\{\alpha^{\mathrm{B}}_m, \alpha^{\mathrm{B}}_{m'_m}\}) < 6$. In lines 16 and 17, the results of the combination of circuits are being recorded while the original circuits are deleted in lines 18 and 19. Line 20 ensures that all modified circuits are combined once in each iteration.

The third process from lines 24 to 26 involves modifying the circuits so that all of them pass by at least one bus terminal. The constraint $d^{V^{\mathrm{B}\prime}}(\alpha_m) > 0$ ensures that the circuits passes by at least one bus terminal. Since buses are parked in bus terminals, it ensures that their circuits have proper start/end points.

In the fourth process from lines 27 onwards, we remove bus routes that are considered to be insignificant in demand with the constraint that all bus stops are served by at least one of the remaining bus routes. The purpose of doing so is to remove bus routes that are not required by a significant number of commuters as they are seen as noise similar to why not every OD pair are considered in the first part of the algorithm. Doing so serves to concentrate resources on bus routes that have high demand. The constraint $m \subseteq m' \mid \forall \max(P^{\mathrm{B}\prime}_m) \geq \frac{1}{2}\max(P^{\mathrm{B}\prime}_1)$ indicates that all circuits with significant amount of demand must be included in the finalized set of bus routes. The constraint $c(G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, \{\alpha^{\mathrm{B}}_{m'}\})) = 1$ indicates that the finalized set of bus routes induces a graph that consists of exactly one component which implies that all bus stops are served by at least one bus route.

6.5 Algorithm Part 3

In the final part of the algorithm, we attempt to approximate the best travel route for all commuters and the bus assignment on each bus route. The method used here assumes that commuters travel in the shortest possible path while ignoring bus waiting time in order to simplify the complex evaluation process. While considering the edges in all bus routes α_m^{B} and the walking path edges in E^{w}, we compute the shortest path for each OD pair. Note that here graph G^{B} is updated to be equal to $G^{\mathrm{B}} = (V^{\mathrm{B}}, \{\alpha^{\mathrm{B}}, E^{\mathrm{w}}\})$ since only edges in walking path and bus routes edges can be considered.

We then remove walking paths before we compute what buses commuters have to take. Let each $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j) \subseteq \gamma^e(G^{\mathrm{B}}, i, j)$ be a set of edges inducing a subcomponent of the shortest path with edges in E^{w} removed, i.e. each component of subgraph induced by the edges in $\gamma^e(G^{\mathrm{B}}, i, j) \setminus E^{\mathrm{w}}$. As we remove the walking path, it can result in a number of disjoint paths consisting only of road edges. Each $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j); q = 1, 2, \dots$ are a set of edges corresponding to each of the components in the set of disjoint paths when edges in E^{w} are removed from $\gamma^e(G^{\mathrm{B}}, i, j)$. Let $\gamma_q^{\mathrm{vr}}(G^{\mathrm{B}}, i, j)$ be the start vertice of subgraph induced by $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$. Let $d(v_i^{\mathrm{B}}, G^{\mathrm{B}})$ be a function that returns the out-degree of v_i^{B} in graph G^{B}.

Let N^{B} define the total number of buses available, while each $n_m^{\mathrm{B}} \in n^{\mathrm{B}}$ is the number of buses assigned to each bus route α_m^{B}. Let $t_{i,j}^{\mathrm{B}}$ be the set of indices of bus routes in α that describes the bus routes a commuter have to board to travel from bus stop v_i^{B} to v_j^{B}. Let $P_m^{\mathrm{B}\prime\prime}$ be the demand matrix for each bus route α_m^{B} and d_m is the demand on each bus route. Let $\tau_{i,j}^{\mathrm{B}\prime}$ be the bus travel time for commuter travelling from v_i^{B} to v_j^{B}. Let $\tau_{i,j}^{\mathrm{B}\prime\prime}$ be the walking time for commuter travelling from v_i^{B} to v_j^{B}. Let β be a variable that can take any value. Let e' be a set of edges. In addition, recall that $V(\alpha_m'')$ is the set of vertices incident to edges in α_m'' and that $D(\alpha_k, V_j)_{\mathrm{in}}$ and $D(\alpha_k, V_j)_{\mathrm{out}}$ are the number of incoming and outgoing edge incident to vertex V_j in subgraph formed with edges in α_k respectively; where α_m'' and α_k are sets of edges.

The third part of the algorithm is as follows.

```
   input: $\alpha^{\mathrm{B}}$, $P^{\mathrm{B}}$,$V^{\mathrm{B}}$,$E^{\mathrm{w}}$
   output: $n^{\mathrm{B}}$,$t^{\mathrm{B}}$,$\tau^{\mathrm{B}\prime}$,$\tau^{\mathrm{B}\prime\prime}$
1: Update $G^{\mathrm{B}} \leftarrow (V^{\mathrm{B}}, \{\alpha^{\mathrm{B}}, E^{\mathrm{w}}\})$
2: Initiate $P^{\mathrm{B}\prime\prime}$ where all elements are 0
```

```
3: for each OD pair $\forall i, j$ do
4:     Initiate $t^{\mathrm{B}}_{i,j} \leftarrow \emptyset$
5:     Initiate $\tau^{\mathrm{B}\prime}_{i,j} \leftarrow 0$
6:     Find $\gamma^{e}(G^{\mathrm{B}}, i, j)$ using Dijkstra's algorithm : $\min \left( \sum_{e^{\mathrm{B}}_{i,j} \in \gamma^{e}(G^{\mathrm{B}},i,j)} c^{\mathrm{B}}_{i,j} \right)$
       Subject to:
       $D^{+/-}(d_k) = 1$
       $D^{+/-}(o_k) = -1$
       $D^{+/-}(v^{\mathrm{B}}_i) = 0 \ \forall v^{\mathrm{B}}_i \in V^{\mathrm{B}} \setminus \{o_k, d_k\}$
7:     $\gamma^{v}(G^{\mathrm{B}}, i, j) \leftarrow V(\gamma^{e}(G^{\mathrm{B}}, i, j))$
8:     for each component $q$ in $G' = (\gamma^{v}(G^{\mathrm{B}}, i, j), \gamma^{e}(G^{\mathrm{B}}, i, j) \setminus E^{\mathrm{w}})$ do
9:         Find $\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j)$ the set of edges in component $q$
10:        Find $\gamma^{\mathrm{vr}}_{q}(G^{\mathrm{B}}, i, j) \leftarrow v^{\mathrm{B}}_i : D(\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j), v^{\mathrm{B}}_i)_{\mathrm{in}} = 0;$
           $D(\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j), v^{\mathrm{B}}_i)_{\mathrm{out}} = 1$
11:    end for
12:    $\tau^{\mathrm{B}\prime\prime}_{i,j} \leftarrow \sum_{i',j' : e^{\mathrm{w}}_{i',j'} \in \gamma^{e}(G^{\mathrm{B}},i,j)} c^{\mathrm{w}}_{i',j'}$
13:    for each $q$ in $\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j)$ do
14:        while $\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j) \neq \emptyset$ do
15:            Find $\gamma^{\mathrm{vr}}_{q}(G^{\mathrm{B}}, i, j) \leftarrow v^{\mathrm{B}}_i : D(\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j), v^{\mathrm{B}}_i)_{\mathrm{in}} = 0;$
               $D(\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j), v^{\mathrm{B}}_i)_{\mathrm{out}} = 1$
16:            Find $m : \max(|e'|)$
               Subject to:
               $e' \subseteq \{\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j) \cap \alpha^{\mathrm{B}}_{m}\}$
               $c^{\min}(G^{\mathrm{B}\prime} = (V^{\mathrm{B}}, e')) = 1$
               $d(\gamma^{\mathrm{vr}}_{q}(G^{\mathrm{B}}, i, j), G^{\mathrm{B}\prime}) = 1$
17:            Update $\gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j) \leftarrow \gamma^{\mathrm{er}}_{q}(G^{\mathrm{B}}, i, j) \setminus e'$
18:            $\tau^{\mathrm{B}\prime}_{i,j} \leftarrow \sum_{i',j' : e^{\mathrm{r}}_{i',j'} \in e'} c^{\mathrm{r}}_{i',j'}$
19:            $t^{\mathrm{B}}_{i,j} \leftarrow \{t^{\mathrm{B}}_{i,j}, m\}$
20:            for each $i', j' : e^{\mathrm{r}}_{i',j'} \in e'$ do
21:                $P^{\mathrm{B}\prime\prime}_{m,i',j'} \leftarrow P^{\mathrm{B}\prime\prime}_{m,i',j'} + P^{\mathrm{B}}_{i,j}$
22:            end for
23:            $d_m \leftarrow \max(P^{\mathrm{B}\prime\prime}_{m})$[20]
```

[20]Note that this equation is $d_m \leftarrow d_m + P^{\mathrm{B}}_{i,j}$ for the Mandl Benchmark since it is assumed

```
24:         end while
25:     end for
26: end for
27: for each m do
28:     $d_m \leftarrow d_m^\beta$
29: end for
30: for each m do
31:     $n_m^{\mathrm{B}} \leftarrow \left\lfloor N^{\mathrm{B}} d_m / \sum_m d_m + 0.5 \right\rfloor$
32: end for
```

In lines 6 to 11, we find $\gamma^v(G^{\mathrm{B}}, i, j)$ $\gamma^e(G^{\mathrm{B}}, i, j)$ $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$ and $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$ for each OD pair. Recall in particular that $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$ is the start vertice of subgraph induced by $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$; and the induced subgraph is essentially the shortest path from node v_i^{B} to v_j^{B} that is why $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$ has an out-degree of 1 while its in-degree is 0.

In line 12 we compute the total walking time after finding the shortest path for each OD pair. The condition in lines 13 and 14 is equivalent to saying for each component and while the component is not empty, do the following. The loops from lines 13 to 25 determine what buses commuters for each OD pair takes. In each iteration in line 15, we find the starting node $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$ of the path described by edges in $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$. We need to find $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$ at each iteration since the starting changes as we remove edges from $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$. In line 16, we find the most suitable bus route α_m^{B} such that it has the most number of edges that coincide with $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$ given that the edges forms a continuous path starting from the start node $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$. This condition is ensured by three constraints in line 16, the first constraint indicates that e' must be a subset of the edges in bus route α_m^{B} and the edges in $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$. The second constraint indicates that the edges in e' must induce a graph of single component, i.e. continuous paths. The third constraint indicates that the out-degree of the starting node $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$ is 1, this implies that the path in the graph induced by the edges in e' starts from the starting node $\gamma_q^{v\mathrm{r}}(G^{\mathrm{B}}, i, j)$ since the starting node is incident to an edge in the subgraph induced by edges in e'.

that commuter can board the first bus that arrives and the need to wait for more than one bus is not taken into account. Therefore it is also necessary to initiate $d_m = 0$ at the start of the algorithm.

After having found the desired bus route α_m^{B}, in line 17 we remove the edges in e' from $\gamma_q^{\mathrm{er}}(G^{\mathrm{B}}, i, j)$ in preparation for the next iteration. In line 18 we update the total in-vehicle travel time. In line 19, we update the bus routes taken so far to keep track of transfers etc. In lines 20 to 22, we update each element of the travel demand matrix $P_m^{\mathrm{B}\prime\prime}$ that is defined for each bus route α_m^{B}. We do this based on the OD pair travel demand and the edges in e'. $P_m^{\mathrm{B}\prime\prime}$ gives us an idea of how many commuters are expected to travel on which bus route along which edges. In line 23, we update d_m which keeps track of the maximum demand in each bus route α_m^{B} along all edges. The rationale of keeping track of the maximum demand is due to the fact that buses have limited capacity and it is necessary to compute the number of buses allocated to each bus route based on this information.

In lines 27 to 29, we smoothen the values of d_m so that the difference in the number of buses allocated to each bus route is not too high. It is found that using the value of $\beta = 0.65$ provides reasonable performance in bus assignments. From line 30 onwards, we simply assign buses to each bus route based on the equation in line 31.

The number of transfers commuters need to make is simply $|t_{i,j}^{\mathrm{B}}| - 1$. The bus wait time for each bus route can be found using the equation

$$w_m = \sum_{\forall i,j : e_{i,j}^{\mathrm{r}} \in \alpha_m^{\mathrm{B}}} \frac{c_{i,j}^{\mathrm{r}}}{2 n_m^{\mathrm{B}}}.$$

The typical convention for bus waiting time is to calculate the half of the period at which buses arrives at each bus stop [23,27]. The total bus waiting time for commuters of each OD pair corresponding to the demand $p_{i,j}^{\mathrm{B}}$ is

$$w_{i,j} = \sum_{m \in t_{i,j}} w_m.$$

This total bus waiting time is commonly known as the out-of-vehicle time. The time spent travelling on a bus $\tau_{i,j}^{\mathrm{B}\prime}$ is commonly known as the in-vehicle time.

6.6 Bus Algorithm Time Complexity

In part 1 of the algorithm, each OD pair is considered and therefore they contribute to a complexity of $\mathrm{O}(|V^{\mathrm{B}}|^2)$. The computation of shortest circuit

$O(2^{|E'|})$ where E' represents the edges that are near to the edges that must be included in the shortest circuit. The indication of near here is considered in terms of the cost. Therefore, the time complexity of the first part of the algorithm is $\mathrm{O}(2^{|E'|}\ |V^{\mathrm{B}}|^2)$. In part 2 of the algorithm, we consider joining each circuits therefore the time complexity here is $\mathrm{O}(|V^{\mathrm{B}}|^4)$ assuming that there are $\mathrm{O}(|V^{\mathrm{B}}|^2)$ number of bus routes. In part 3 of the algorithm, we consider the application of each bus routes for each OD pair therefore the complexity is also $\mathrm{O}(|V^{\mathrm{B}}|^4)$ assuming that there are $\mathrm{O}(|V^{\mathrm{B}}|^2)$ number of bus routes.

As the value $|E'|$ is typically small, therefore the overall time complexity of the bus algorithm is $\mathrm{O}(|V^{\mathrm{B}}|^4)$. Despite the seemingly high complexity, many of the computations can be done in parallel to reduce the amount of time taken. In addition, the actual number of circuits used in the computations are typically low due to the pruning away of insignificant OD pairs in part 1 of the algorithm and the pruning away of bus routes with insignificant demand at the end of part 2 of the algorithm.

6.7 Bus Route Simulation Setup (NUS)

A simulation was carried out for the NUS environment and compared against the results against the existing NUS bus routes. Most of the data used are results of a number of estimations and assumptions made; details of these estimations and assumptions will be explained in the following paragraphs. Real data could be useful in future work but is not currently available. In figure 6.1, we can see the original NUS map and each bus stop is given a corresponding number. In figure 6.2, we can see a digitally drawn version. Each edge represents the shortest path from one bus stop to another. A number representing the amount of time (in minutes) it takes to traverse the edge is shown next to each edge near the arrowheads. Amongst all bus stops, bus stops 1 and 14 are bus terminals. Note that edges between nodes 1 and 7 and nodes 3 and 5 traverses Clementi Road.

A number of estimations and assumptions are made. Firstly, we make estimations in the amount of time taken to traverse the roads along the shortest path between each bus stop. The estimations made was based on travel experiences and was found to be very similar to the distance of each road divided by 20 km/hr, i.e. the travel experiences was found to be consistent to the shuttle buses travelling at 20 km/hr. As Clementi Road is currently

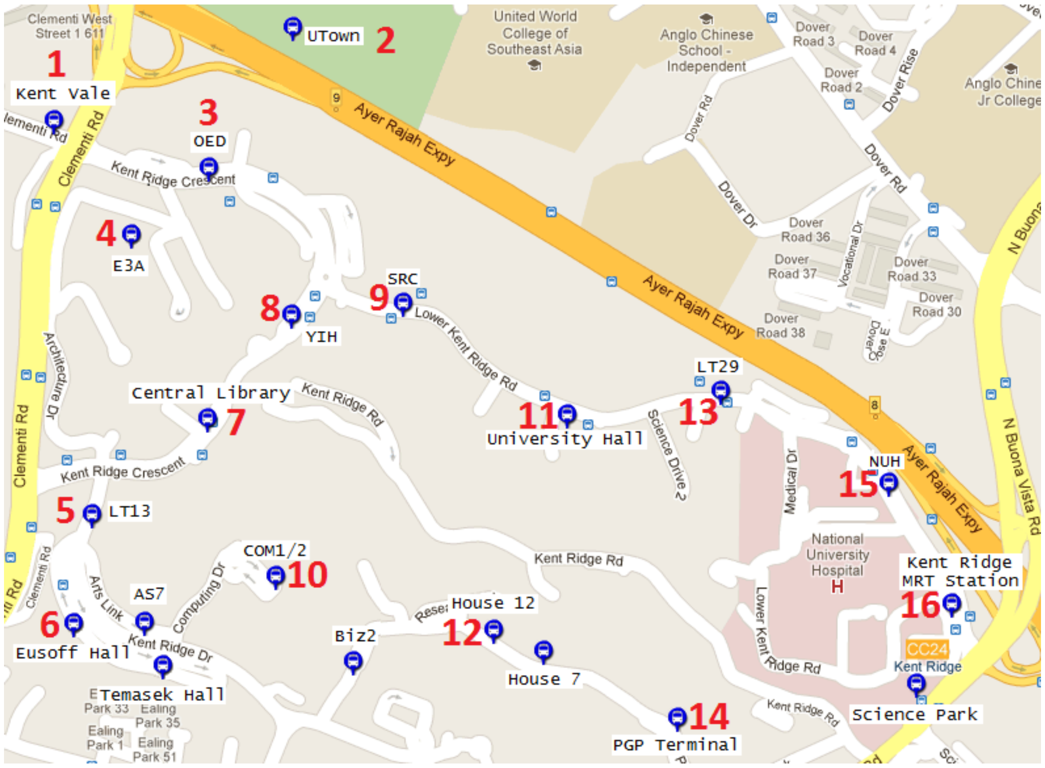


Figure 6.1: NUS Map with Bus Stops. All bus stops are numbered. Bus stop 1 and 14 are bus terminals.

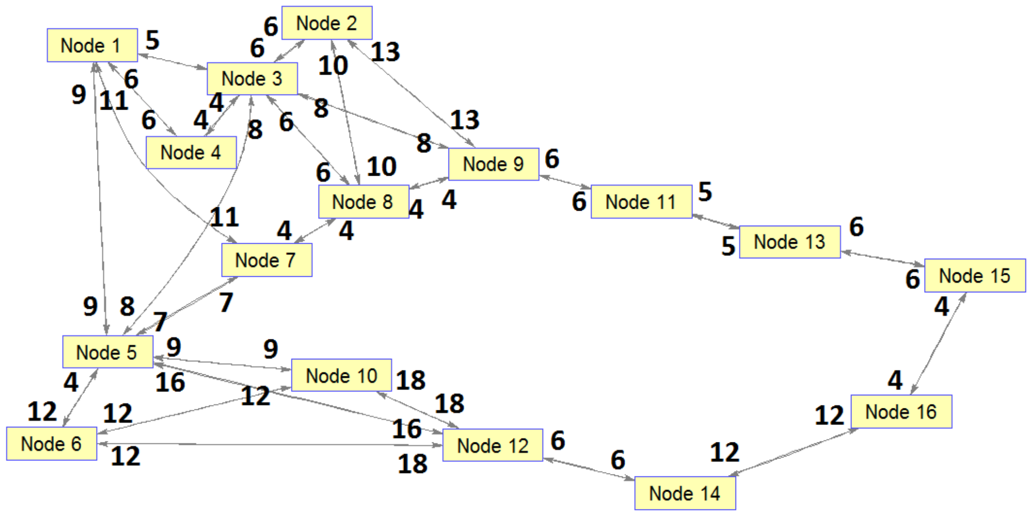


Figure 6.2: Digitally Drawn Version of NUS Map with Bus Stops. Each edge represents the shortest path from one bus stop to another. A number representing the amount of time (in minutes) it takes to traverse the edge is shown next to each edge near the arrowheads.

not traversed by any bus service, it is assumed that buses travel at 60 km/hr as it is not within NUS campus area and driving at 20 km/hr is too slow. The time taken to traverse the edges along the shortest path from one bus stop to another is shown in figure 6.2. With these information, we can form the matrix E^{r} and $c^{\mathrm{r}}_{i,j}$ the amount of time taken to traverse each edge $e^{\mathrm{w}}_{i,j}$.

Secondly, we estimate the amount of time it takes to walk between each bus stops. We assume a walking speed of 4.4 km/hr. The distances between bus stops are taken to be the measured distance along the most commonly known path between them. The amount of time taken to travel between bus stops can be seen in table 6.1.

Thirdly, the demand matrix P^{B} shown in table 6.2 is estimated based on a manually conducted survey by counting the number of people at each bus stops and the number of people who board/alight each bus services. The demand matrix is then modified based on feedbacks given by some bus drivers so that it is be more representative of their personal experiences. After the modification, some noise was added so that there is no zero values in the matrix P^{B} and to add some randomness in the simulation. The noise is pre-calculated and stored so that each time the simulation is ran, the amount of

-	-	10	12	18	-	-	18	-	-	-	-	-	-	-	-
-	-	12	18	-	-	-	-	-	-	-	-	-	-	-	-
10	12	-	6	16	-	16	8	16	-	-	-	-	-	-	-
12	18	6	-	-	-	12	14	-	-	-	-	-	-	-	-
18	-	16	-	-	8	8	16	-	18	-	-	-	-	-	-
-	-	-	-	8	-	16	-	-	-	-	-	-	-	-	-
-	-	16	12	8	16	-	8	16	10	-	-	-	-	-	-
18	-	8	14	16	-	8	-	8	18	-	-	-	-	-	-
-	-	16	-	-	-	16	8	-	-	12	-	-	-	-	-
-	-	-	-	18	-	10	18	-	-	-	12	-	-	-	-
-	-	-	-	-	-	-	-	12	-	-	14	10	-	-	-
-	-	-	-	-	-	-	-	-	12	14	-	-	12	-	-
-	-	-	-	-	-	-	-	-	-	10	-	-	-	12	-
-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	8
-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-

Table 6.1: Walking Time Matrix. Each $c^{\mathrm{w}}_{i,j}$ represents the amount of time taken to walk from bus stop v^{B}_i to v^{B}_j. Elements with no value indicates that the walking time required can be derived by solving for their shortest path in the graph $G^{\mathrm{w}} = (V^{\mathrm{B}}, E^{\mathrm{w}})$

noise added is the same for each element.

6.8 Simulation Results (NUS)

In the simulation carried out, a set of bus routes α^{B} was computed based on the algorithm in section 6.2 and data described in section 6.7. The bus routes were computed in 47 seconds on a 2 GHz computer. In addition, we apply the third part of the algorithm of section 6.2 to the NUS bus routes to determine the travel time for commuters. Let $\tau^{\mathrm{BN}\prime}_{i,j}$,$\tau^{\mathrm{BN}\prime\prime}_{i,j}$,$w^{N}_{i,j}$ be the total in-vehicle travel time, total walking time, and total bus waiting time for each commuter travelling on the NUS bus routes from bus stop v^{B}_i to v^{B}_j respectively; and let $\tau^{\mathrm{BP}\prime}_{i,j}$,$\tau^{\mathrm{BP}\prime\prime}_{i,j}$,$w^{NP}_{i,j}$ be the ones for commuter travelling on the bus routes computed by the proposed algorithm. Some of the results are

0	5	3	13	5	1	1	1	5	9	3	5	4	5	8	10
5	0	8	15	4	3	23	16	5	19	4	5	14	3	5	3
3	8	0	11	8	6	10	6	4	9	3	1	8	3	2	3
13	15	11	0	13	17	48	41	7	37	11	10	37	19	6	19
5	4	8	13	0	6	9	6	3	3	7	9	13	3	7	2
1	3	6	17	6	0	16	16	5	18	5	5	14	5	5	6
1	23	10	48	9	16	0	37	8	48	10	10	48	17	4	19
1	16	6	41	6	16	37	0	6	42	4	2	40	17	5	15
5	5	4	7	3	5	8	6	0	4	4	4	8	7	4	6
9	19	9	37	3	18	48	42	4	0	6	10	44	15	10	12
3	4	3	11	7	5	10	4	4	6	0	5	8	9	9	3
5	5	1	10	9	5	10	2	4	10	5	0	7	5	3	6
4	14	8	37	13	14	48	40	8	44	8	7	0	21	9	16
5	3	3	19	3	5	17	17	7	15	9	5	21	0	5	4
8	5	2	6	7	5	4	5	4	10	9	3	9	5	0	1
10	3	3	19	2	6	19	15	6	12	3	6	16	4	1	0

Table 6.2: Demand Matrix P^{B}. Each $p^{\mathrm{B}}_{i,j}$ represents the number of commuters who wish to travel from bus stop v^{B}_i to v^{B}_j

as follows:

$$\begin{array}{ll} \sum_{\forall i,j} p^{\mathrm{B}}_{i,j}\left(\tau^{\mathrm{BN}\prime}_{i,j}\right) = 48861 & \sum_{\forall i,j} p^{\mathrm{B}}_{i,j}\left(\tau^{\mathrm{BN}\prime\prime}_{i,j}\right) = 7480 \\ \sum_{\forall i,j} p^{\mathrm{B}}_{i,j}\left(\tau^{\mathrm{BP}\prime}_{i,j}\right) = 41788 & \sum_{\forall i,j} p^{\mathrm{B}}_{i,j}\left(\tau^{\mathrm{BP}\prime\prime}_{i,j}\right) = 8860 \end{array} .$$

We can see that the computed bus route requires commuters to travel by walking for an additional time of $8860 - 7480 = 1380$ minutes. As the total number of commuters is

$$\sum_{\forall i,j} p^{\mathrm{B}}_{i,j} = 2568 \quad ,$$

each commuter on average have to walk for $1380/2568 = 0.537$. Therefore, on average each commuter has to walk about half a minute more than the situation where the NUS bus routes were used instead. The simulation results indicate that regardless of the total number of bus N^{B}, bus waiting time is always lesser when we use NUS bus routes. In figure 6.3, we see a plot of the difference in total bus wait time. When the total number of bus is 10,

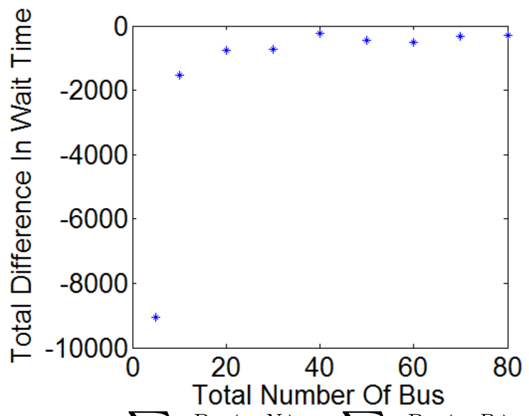


Figure 6.3: Plot Of $\sum_{\forall i,j} p^{\mathrm{B}}_{i,j}\left(w^{N}_{i,j}\right) - \sum_{\forall i,j} p^{\mathrm{B}}_{i,j}\left(w^{P}_{i,j}\right)$ Against N^{B}

NUS bus routes results in a total bus wait time that is lesser by 763 minutes. Therefore on average, each commuter waits $763/2568 = 0.297$ minutes more for a bus if we apply the computed bus route; this is about 18 seconds more.

Since assuming that the total number of shuttle bus in NUS is 5 is much too low an estimation, we assume that the total number of shuttle bus is 10, i.e. $N^{\mathrm{B}} = 10$. With this assumption, we can calculate the total amount of time saved by the computed bus routes

$$\sum_{\forall i,j} p^{\mathrm{B}}_{i,j}\left(\tau^{\mathrm{BN}\prime}_{i,j} + \tau^{\mathrm{BN}\prime\prime}_{i,j} + w^{N}_{i,j} - \tau^{\mathrm{BP}\prime}_{i,j} - \tau^{\mathrm{BP}\prime\prime}_{i,j} - w^{P}_{i,j}\right) = 4930 \quad .$$

As we can see from the calculation, the total travel time is lesser by 4,930 minutes for all commuters if the computed bus routes are used. On average, each commuters enjoys a reduction of total travel time of $4930/2568 = 1.92$; this is about 2 minutes. Note that since the demand matrix P only correspond to a single trip, the average calculated is for each journey for each commuter. This implies that if a commuter travels around NUS more than one time, the amount of time saved is increased accordingly. Another advantage of the computed bus routes is that it is easier to implement as there are only three bus routes to consider; the NUS bus routes however have 6 bus routes to consider (refer to table 6.3).

The NUS bus routes and the computed bus routes are shown in figures 6.4 and 6.5 respectively. All edges in both figures are bidirection except for the ones with an arrow head attached to them; buses only travel in the direction pointed by the arrow head. Note that NUS bus route pair D1, D2 are combined in the figure 6.4 since most of their paths overlap. Also note that despite the fact that the arrow head is not shown for NUS bus route A, it is in fact two bus routes; bus route A1 travels anti-clockwise in the edges of “circuit A” shown in figure 6.4 while A2 goes clockwise.

The presence of edges that can only be traversed in a single direction implies that the bus routes planned lacks symmetry therefore this reflects a minor problem in the NUS bus routes. In addition, bus route D1 and D2 does not visit any of the two bus terminals (bus stops 1 and 14). This is another minor problem as explicit efforts have to be made to travel to their stating bus stop when they start the day, and they have to make additional journey back to the bus terminal when they end the day. The node sequence is shown in table 6.3.

Node Sequence	
NUS A1	[14,16,15,13,11,9,8,7,5,10,12,14]
NUS A2	[14,12,10,5,7,8,9,11,13,15,16,14]
NUS B	[1,4,3,8,7,5,10,6,5,7,8,3,4,1]
NUS C	[1,4,3,9,11,13,15,13,11,9,3,4,1]
NUS D1	[12,10,5,7,8,3,2,9,11,13,15]
NUS D2	[15,13,11,9,3,2,8,7,5,10,12]
Computed Bus Route 1	[1,4,3,5,7,8,9,11,13,15,16,15,13,11,9,8,7,5,3,4,1]
Computed Bus Route 2	[1,4,3,5,12,14,16,15,13,11,9,8,7,8,9,11,13,15,16,14,12,5,3,4,1]
Computed Bus Route 3	[1,7,8,2,3,5,10,6,10,5,3,2,8,7,1]

Table 6.3: Node Sequence Of Bus Routes For NUS Simulation

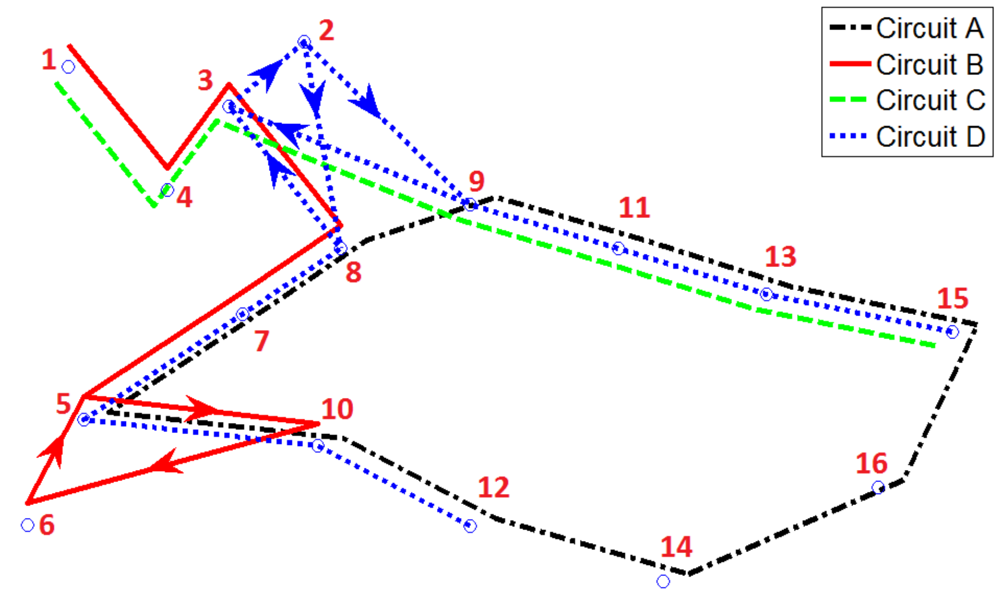


Figure 6.4: NUS Bus Routes

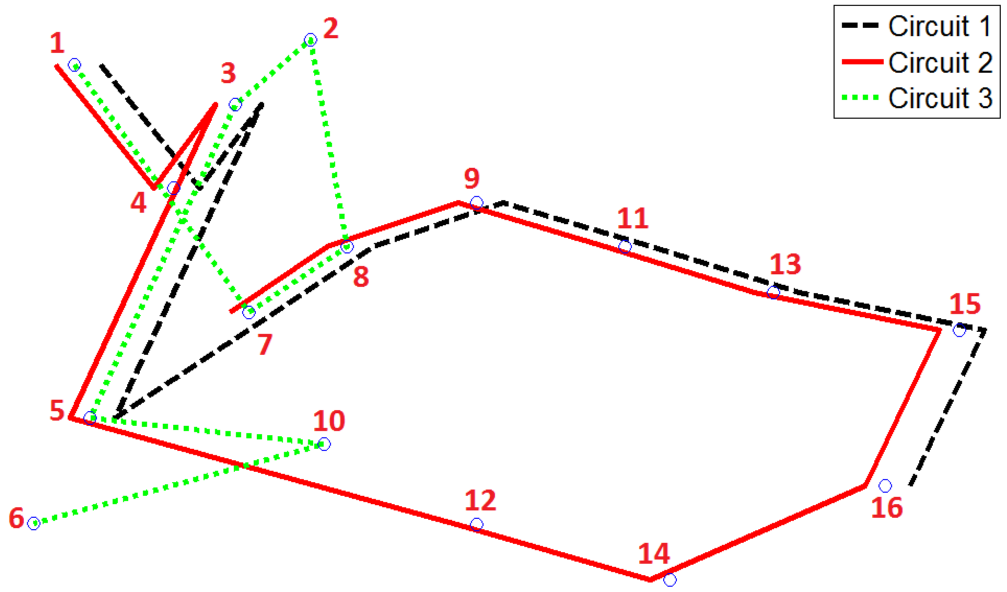


Figure 6.5: Computed Bus Routes

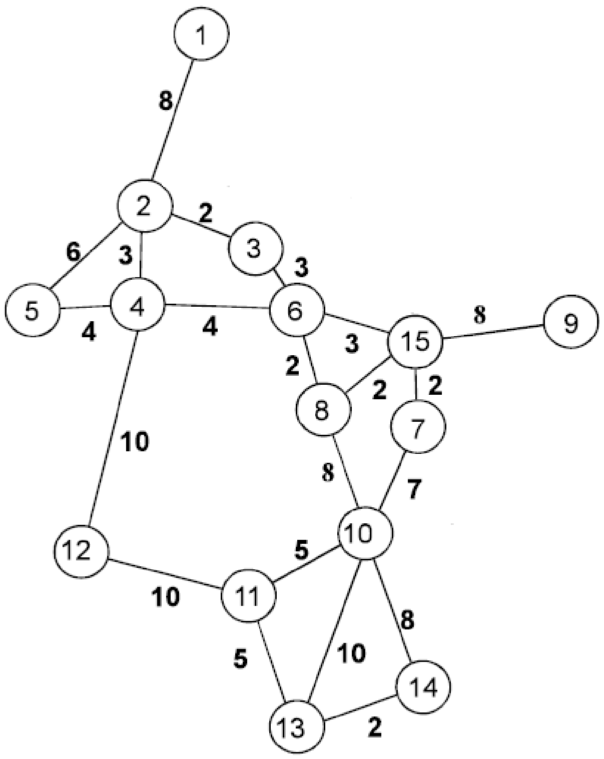


Figure 6.6: The road network of Mandl benchmark. Time taken (minutes) to travel between each node is displayed in bold.

6.9 Mandl's Benchmark

In the Mandl benchmark, there are a total of 15,570 commuters and 15 nodes each representing a bus stop. The concept of bus terminal is not introduced and commuters are not allowed to walk between bus stops [22,23]. The benchmark is made so that performance of bus route algorithms can be compared in a fair manner. The network and demand matrix can be seen in figure 6.6 and table 6.4 respectively. The performance of various solutions on the Mandl benchmark is shown in Table 6.5.

In table 6.5, the in-vehicle time is the amount of time spent travelling while in a bus. The out-of-vehicle time is the amount of time spent waiting for a bus either for the first bus or subsequent transfers. The transfer time is a penalty of 5 minutes applied for each transfer made by each person. The travel time is the sum of these three time values.

In Shih [20], two solutions are proposed. The coordinated version im-

0	400	200	60	80	150	75	75	30	160	30	25	35	0	0
400	0	50	120	20	180	90	90	15	130	20	10	10	5	0
200	50	0	40	60	180	90	90	15	45	20	10	10	5	0
60	120	40	0	50	100	50	50	15	240	40	25	10	5	0
80	20	60	50	0	50	25	25	10	120	20	15	5	0	0
150	180	180	100	50	0	100	100	30	880	60	15	15	10	0
75	90	90	50	25	100	0	50	15	440	35	10	10	5	0
75	90	90	50	25	100	50	0	15	440	35	10	10	5	0
30	15	15	15	10	30	15	15	0	140	20	5	0	0	0
160	130	45	240	120	880	440	440	140	0	600	250	500	200	0
30	20	20	40	20	60	35	35	20	600	0	75	95	15	0
25	10	10	25	15	15	10	10	5	250	75	0	70	0	0
35	10	10	10	5	15	10	10	0	500	95	70	0	45	0
0	5	5	5	0	10	5	5	0	200	15	0	45	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6.4: Travel demand matrix of Mandl benchmark. It is symmetrical.

Solution	No. of bus	No. of route	% of commuters			Total time			
			Zero transfer	One transfer	Two transfer	Travel time	In vehicle time	Out of vehicle time	Transfer time
Shih 1998 (Coordinated)	87	6	82.59	17.41	0	225,102	191,826	19,726	13,550
Shih 1998 (Uncoordinated)	84	6	82.59	17.41	0	203,936	170,328	20,058	13,550
Baaj 1995	82	7	80.99	19.01	0	217,954	180,356	22,804	14,800
Mandl 1980	99	4	69.94	29.93	0.13	219,094	177,400	18,194	23,500
Saeed 2011	87	12	83.66	15.21	0.95	205,109	167,198	24,591	13,320
Alt 2011 (Current Best)	87	4	86.38	10.79	2.83	196,520	156,110	27,610	12,800
Proposed Algorithm	87	5	85.74	12.14	2.12	201,712	161,690	27,272	12,750
Proposed Algorithm	82	5	85.74	12.14	2.12	203,346	161,690	28,906	12,750

Table 6.5: Performance Comparison Using Mandl Benchmark.

plies that departure times are modified to minimize transfer times, while the uncoordinated version means that such modification is not made. All other algorithms including the one proposed in this thesis did not make such modification.

One can easily verify transfer time penalty by doing very simple calculations. For example, the proposed algorithm have 12.14% commuter making 1 transfer and 2.12% commuter making 2 transfers therefore its penalty is $(12.14 + 2 \times 2.12)/100 \times 15570 \times 5 = 12750$. For Mandl [23], $(29.93 + 2 \times 0.13)/100 \times 15570 \times 5 = 23500$. The same can be seen in Shih, Baaj and Alt [20, 21, 28] as well. However, there seems to be a mistake in the result presented in Saeed [24]. Its true transfer time penalty should be $(15.21 + 2 \times 0.95)/100 \times 15570 \times 5 = 13320$. Saeed has reported a penalty of 10465 instead in [24]. The mistake is already corrected in the results shown in table 6.5.

The solution in Alt [28] is performed on the Mandl bechmark but the evaluation method is modified and is different from the conventional means. Therefore, I have taken the bus route it has calculated in [28] and evaluated its performance using part 3 of the proposed algorithm from section 6.5. The bus route computed in [28] is believed to be the current state-of-the-art in literature as it has the best results in terms of Total Travel Time in the Mandl benchmark. In addition, the reported bus route is the best out of several attempts in which differences in bus route is noted in each attempt according to [28]. This is not surprising since the method adopted is stochastic.

Upon comparison, it can be seen that the proposed algorithm is superior to all other solution except for the one presented in [28]. Comparing the transfer time, one can see that the proposed algorithm have lesser transfer made compared to solution from [28]. In fact, the proposed solution requires the least amount of transfer out of all solutions. However, in terms of Total Travel Time, the solution in [28] is 3.4% lower than that of the proposed solution. Therefore, the proposed algorithm is slightly inferior to the state-of-the-art.

Based on the computation on Mandl's benchmark, the proposed algorithm implemented on Matlab was able to compute the bus routes in 40 seconds on a 2 GHz computer. The algorithm in [28] implemented on the "fast prototyping language Python" was only able to complete the computation in "about one hour on a 2 GHz computer" (quoted from [28]). Therefore, even though the state-of-the-art algorithm was able to perform better than the proposed algorithm in this thesis by 3.4% in terms of total travel time it requires a significantly higher amount of time to compute its bus routes for the Mandl's benchmark.

6.10 Summary

In this chapter, we identified the similarities between the bus route planning problem and the probe allocation problem. We also introduced a novel algorithm which was a modified version of the fixed circuit algorithm. A NUS simulation was carried out and it was found that the proposed algorithm was able to compute a bus route that performs better than the existing NUS bus routes. In addition, the proposed algorithm was applied to the Mandl's benchmark and it was found that it outperforms most existing algorithms and that while it was slightly inferior to the state-of-the-art algorithm, it was significantly faster in terms of computational time required.

Chapter 7

Conclusions And Future Work

7.1 Conclusion

In this thesis, the novel probe allocation problem was introduced and solved using the proposed algorithms. A special case was identified and its similarities with the Chinese Postman Problem were noted. Two different algorithms were proposed, one for undirected graphs (the algorithm is called modified Chinese postman algorithm) and another for directed graphs (the relaxed eulerian circuit Algorithm). The algorithms run in polynomial time and produce optimum solutions. For the probe allocation problem in general case, two categories of algorithms were proposed, the first was the fixed circuit algorithm and the other was the random walk algorithm.

A Singapore scale simulation was carried out using the proposed algorithms, together with the data provided by LTA and ComfortDelgro which amounts to approximately 31GB of data in total. The approximate fixed circuit algorithm took the longest amount of time requiring about 4 hours and 45 minutes on a 2.4 GHz computer. The computation also requires the machine to have at least 4GB RAM. The shortest amount of time is required by the random walk algorithms requiring about 5 minutes.

According to the simulation results, it was found that 36,669 agents are required for the fixed circuit solution in order to make accurate traffic speed estimations in the Singapore road network which consists of 77,283 roads. However, if we were to install stationary sensors on road that needs to be visited frequently, i.e. require agents to visit them once every 30 minutes or lesser (a total of 12,538 out of 77,283 roads in Singapore), the number of agents required is reduced to 1,151. Therefore it means that we can install 12,538 sensors on roads and deploy 1,151 agents to make accurate traffic speed estimations.

In the case for the random walk without communication algorithm, it was found that 327,912 agents are required. Assuming that agent distribution is similar to that of the cars in Singapore, we can consider installing sensors on 327,912 private cars which is about 57.5% of all private cars in Singapore. However, if we install the 12,538 stationary sensors on roads that needs to be visited frequently it was found that we only need to install sensors on 189,004 private cars which is only 33.11% of private cars.

Finally, similarities between the probe allocation problem and the bus route planning problem were identified. The fixed circuit algorithm was modified so that it is applicable to the bus route planning problem. It was found that the bus routes computed by the proposed algorithm outperform the NUS bus routes that are currently being used. In addition, the performance of the proposed algorithm was found to outperform many existing bus route planning algorithms when compared against the Mandl benchmark. The performance of the proposed algorithm is slightly inferior in to the state-of-the-art as it produces a total travel time 3.4% higher. However, the proposed algorithm is able to compute the bus routes significantly faster, i.e. the state-of-the-art is reported in [28] to take "about an hour on a 2 GHz computer" to compute its bus routes while the proposed algorithm took only 40 seconds on a 2 GHz computer.

7.2 Future Work

In this section, we discuss some of the possible future work that can be continued from this thesis.

The first of these possibilities would be the "Probe Allocation By Kirchhoff's Circuit Laws". We note that the solutions proposed for the probe allocation problem requires that agents traverse all edges in order to measure the traffic speed along them. However, similar to the principles of Kirchhoff's Circuit Laws, we know that it is not necessary to make measurement along all edges to know the traffic speed along it. With the use of the fundamental diagram, traffic density, flow and speed can be inferred from one another as well. It is believed that the application of these methods can potentially reduce the number of agents required significantly.

Secondly, a Singapore scale simulation can be carried for the bus route planning problem. However, in order to carry out this simulation, it is necessary to know the demand for public transportation for each Origin-Destination pair. This can be easily found out if EZ-link tap card data is available. However, the data is currently not available and therefore simulations cannot be carried out. In addition, the algorithm for the bus route planning problem can be extended to include the consideration of other modes of transportation; such as taking a taxi in middle of trips, MRT and LRT etc. It is believed that this work can potentially improve total travel time for Singaporeans.

Thirdly, a possible future work is to write an efficient algorithm that determines how taxi random walk should be done to ensure high quality of service and maximization of profit. Meaning to say, if we are able to model and predict taxi travel demand as a Markov Chain, we can calculate a suitable transition matrix that determines the random walk for taxi such that it is expected to maximize profit and quality of service. This is analogous to the modification of the fixed circuit algorithm so that it is applicable to the bus route planning problem. Here we modify the random walk algorithm proposed in this thesis and apply it to the taxi random walk problem.

Bibliography

[1] Thulasiraman K., Swamy M. N. S., 1992. "Graphs: Theory and Algorithms." John Wey & Sons, Inc., 1992, Chapter 11, pp. 334-346, ISBN 978-0471513568.

[2] Gwilliam K., 2002. "Cities on the move : a World Bank urban transport strategy review." 2002, ISBN 978-0821351482.

[3] Een N., Srensson N., 2004. "An Extensible SAT-solver." In Theory and Applications of Satisfiability Testing, 2004, Volume 2919, pp. 333-336.

[4] Een N., Srensson N., 2006. "Translating Pseudo-Boolean Constraints into SAT", Journal on Satisfiability, Boolean Modeling and Computation 2(3-4), pp. 1-25.

[5] Bernard M., 2005. "Traffic Congestion: How Predictable? Discovering Volume Trends across Time and Confirming Fundamental Speed-Flow-Density Relations." Independent Research, Princeton University. May 2005.

[6] Aloul F., Al-Rawi B., Al-Farra A., Al-Roh B., 2006. "Solving Employee Timetabling Problems Using Boolean Satisfiability." In The Innovations in Information Technology, 2006 , pp. 1-5.

[7] Ngoduy D., Liu R., "Multiclass first-order simulation model to explain non-linear traffic phenomena", Physica A: Statistical Mechanics and its Applications, Volume 385, Issue 2, 15 November 2007, pp. 667-682.

[8] Donnell E.T., Hines S.C., Mahoney K.M., Porter R.J., McGee H., 2009. "Speed Concepts: Informational Guide." U.S.Department of Trasnportation, Federal Highway Administration, September 2009, Publication No. FHWA-SA-10-001.

[9] Li J., Cheny Q.Y., Wang H., Ni D., 2011. "Analysis of LWR model with fundamental diagram subject to uncertainties." In Transportmetrica, 2011, Volume 0, Issue 0, pp. 1-19.

[10] Jun J., 2010. "Understanding The Variability of Speed Distributions Under Mixed Traffic Conditions Caused by Holiday Traffic." In Transportation Research Part C: Emerging Technologies, Volume 18, Issue 4, August 2010, pp. 599-610.

[11] Department of Statistics Singapore. "Key Annual Indicators." Retrieved April 08, 2012, from http://www.singstat.gov.sg/stats/keyind.html.

[12] Aslam J., Lim S., Pan X., Rus D., 2012. "City-Scale Traffic Forecasting with Roving Sensor Network." In The 10th ACM Conference on Embedded Network Sensor Systems (SenSys12), In Submission, 2012.

[13] Paul N., John K., 2009. "Hidden markov map matching through noise and sparseness." International Conference on Advances in Geographic Information Systems, pp. 336-343

[14] Harold W. K., 1955. "The Hungarian Method for the assignment problem", Naval Research Logistics Quarterly, 2:8397, 1955.

[15] Herbert F., 1991. "X.1 Algorithms for Eulerian Trails", Eulerian Graphs and Related Topics. pp. X.113, ISBN 978-0-444-89110-5.

[16] Andrieu C., Freitas N. D., Doucet A., Jordan M. I., 2003. "An Introduction to MCMC for Machine Learning", Machine Learning, 50, pp. 5-43, 2003.

[17] Tang Y., Liu X., Chen J.H., 2011. "Optimization of pseudo-boolean satisfiability algorithm for FPGA routing ", Information Science and Technology (ICIST), pp. 45-49.

[18] Rintanen J., Heljanko K., Niemela I., 2006. "Planning as satisfiability: parallel plans and algorithms for plan search", Artificial Intelligence, 170, Issues 12-13, pp. 1031-1080.

[19] Lehoucq, R.B., D.C. Sorensen, and C. Yang, 1998. "ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods", SIAM Publications, Philadelphia, 1998.

[20] Shih, M., Mahmassani, H.S., Baaj, M., 1998. “A planning and design model for transit route networks with coordinated operations. transport. In: Transportation Research Record”, Journal of the Transportation Research Board, No. 1623, Transportation Research Board of The National Academies, pp.16-23.

[21] Baaj, M.H., Mahmassani, H.S., 1995. “Hybrid route generation heuristic algorithm for the design of transit networks.” Transportation Research Part C 3 (1), pp. 31-50.

[22] Mandl, C.E., 1979. Applied network optimization. Academic Press, London.

[23] Mandl, C.E., 1979. “Evaluation and optimization of urban public transportation networks.” European Journal of Operational Research 5 (6), pp. 396-404.

[24] Saeed, A.B., Avishai, C., 2011. “Transit-network design methodology for actual-size road networks.” Transportation Research Part B 45(10), pp. 1787-1804

[25] Fang, Z., Albert, G., 2003. “Optimization of Transit Network to Minimize Transfers.” Lehman Center for Transportation Research, Florida International University.

[26] Valerie, G., Jin-Kao, H., 2008. “Transit Network Design And Scheduling: a Global Review.” Transportation Research Part A 42(10), pp. 1251-1273

[27] Silman, L.A., Barzily, Z., Passy, U., 1974. Planning the route system for urban buses. Computer and Operations Research 1 (2), 201-211

[28] Alt, B., Weidmann, U., 2011. “A stochastic multiple area approach for public transport network design.” Public Transport Volume 3, Number 1 (2011), pp. 65-87

[29] Kuan, M.K., 1962. “Graphic programming using odd or even points,” Chinese Math., Vol. 1, 273-277.

[30] Espinasse, T., Gamboa, F., Kien, J-N, Loubes, J.-M., 2012. “Modeling and estimation for Gaussian elds indexed by graphs, application to

road trafc prediction." Interdisciplinary Workshop on Inference, French National Research Agency.

[31] Shunsuke I., 1993. "Information Theory for Continuous Systems." World Scientific, pp. 2, ISBN 978-981-02-0985-8.

[32] Monica B., 2011. "Fundamentals in Information Theory and Coding." Springer, pp. 11, ISBN 978-3-642-20346-6.

[33] Lazo A., Rathie P., 1978. "On the entropy of continuous probability distributions." Information Theory 24(1), pp. 120-122.

[34] Olszewski P., Fan H. S.L., Tan Y.W., 1993. "Area-Wide Traffic Speed-Flow Model For The Singapore CBD." Transportation Research Part A 29(4), pp. 273-281

[35] Vrancken J., dos Santos Soares M., Ottenhof F., 2008. "A real-life test bed for multi-agent monitoring of road network performance." Infrastructure Systems and Services: Building Networks for a Brighter Future (INFRA), 2008 First International Conference on, pp. 1-4, 10-12 November 2008.

[36] Almejalli K., Dahal K., Hossain A., 2009. "An intelligent multi-agent approach for road traffic management systems." Control Applications, (CCA) & Intelligent Control, (ISIC), 2009 IEEE, pp. 825-830, 8-10 July 2009.

[37] Chen F, Pang H, 2008. "Study of multi-agent area coordination control for urban traffic." Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on, pp. 4046-4050, 25-27 June 2008.

[38] Ow Y.X., Chitre M., Rus D., 2011. "Planning for Data Acquisition." Future Urban Mobility IRG, 12-13 January 2011.

[39] Ow Y.X., Chitre M., Rus D., 2012. "Probe Allocation Strategy." Future Urban Mobility Symposium 2012, 11-12 January 2012.

[40] Ow Y.X., Chitre M., Rus D., 2013. "The Probe Allocation Problem." CIVTS 2013, IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems, In Submission, 2013.

[41] Ow Y.X., Chitre M., Rus D., 2013. "An Approximate Bus Route Planning Algorithm." CIVTS 2013, IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems, In Submission, 2013.